

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**



**FEUP** FACULDADE DE ENGENHARIA  
UNIVERSIDADE DO PORTO

# **Multiuser Human-Machine Interface through Augmented Reality**

**João Daniel Ferreira Peixoto**

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

Supervisor: Dr. Marcelo Petry

Co-supervisor: Prof. Dr. António Coelho

July 13, 2023

© João Daniel Ferreira Peixoto, 2023

# **Multiuser Human-Machine Interface through Augmented Reality**

**João Daniel Ferreira Peixoto**

**MASTER IN ELECTRICAL AND COMPUTER ENGINEERING**

Approved in public examination by the Jury:

Chairman: Sérgio Reis Cunha

Arguer: Mauri Ferrandin

July 13, 2023



# Abstract

Extended Reality has become very popular over the last few years, this is due to the expansion on this technology and new applications in multiple fields. This universal paradigm is comprised of concepts like Virtual Reality, Augmented Reality and Mixed Reality - the subject of this dissertation is primarily focused on Augmented Reality (AR). This technology is capable of increasing the user experience by overlaying reality with virtual elements and due to the rapid evolution of the industry, this approach is being ingrained in manufacturing systems.

The aim of this project is to present and assess the main concepts needed to understand the implementation of a multiuser interface for communication between human and machine through the use of Augmented Reality. In order to achieve this, multiple topics were researched: from Augmented Reality and its modern applications to Human-Machine Interaction and how it is employed to optimize collaboration. In-between there are still many other fields needed to be referenced, like the type or AR display devices, how the AR tracking is done and how collaborative robots function. It is also investigated the different type of User Interfaces and how the concept is applied to AR. Additionally, it is also performed research on the implementation of Multiuser Systems through the use of networking solutions in an AR environment.

After assimilating the paradigms investigated throughout the report, it is proposed a methodology to accomplish the goal of the project: With the assistance of an Augmented Reality device, a group of users should be able to grab and control a virtual model of the machine simultaneously. The users should also have the possibility to visualize the statistics of the machinery and its processes due to the overlaying of an interface to the users' field of view. The system's architecture requirements are then broken down into smaller components in order to plan for its implementation, and since the system needs to withstand multiple concurrent users it is also proposed the creation of a multiuser protocol through the employment and adaptation of an open-sourced network library - Mirror. The architecture is planned to be capable of generating multiple application server instances in a single IP address by taking advantage of port multiplexing and a master server that controls and monitors each instance.

Then the actual implementation of the system is described in detail, and how each technology was incorporated into the project. The main challenges that needed to be overcome for the project's success and their respective implemented solutions are exposed and explained. The integration of each component is elucidated in a sequential method by decomposing the application development into multiple scenes. Finally, the project is evaluated to test its hard limits and analyze potential areas of improvement by performing multiple stress tests on the server system. It is also taken into account user feedback on the application, in order to check if it is performing as expected.

**Keywords:** Augmented Reality, Human-Machine Interaction, User Interface, Collaborative Robots, Multiuser Collaboration



# Resumo

A Realidade Estendida tornou-se muito popular nos últimos anos devido à expansão dessa tecnologia e às novas aplicações em vários campos. Esse paradigma universal é composto por conceitos como Realidade Virtual, Realidade Aumentada e Realidade Mista - o foco desta dissertação é principalmente a Realidade Aumentada. Essa tecnologia é capaz de aumentar a experiência do utilizador sobrepondo elementos virtuais à realidade e, devido à rápida evolução da indústria, essa abordagem está a ser incorporada em sistemas de manufatura.

O objetivo deste projeto é apresentar e avaliar os principais conceitos necessários para entender a implementação de uma interface multi-utilizador para comunicação entre humano e máquina por meio do uso da Realidade Aumentada (RA). Para alcançar isso, vários tópicos foram pesquisados: desde a Realidade Aumentada e as suas aplicações mais modernas até a Interação Humano-Máquina e como ela é aplicada para otimizar a experiência de colaboração. Entre esses tópicos, há muitos outros campos que precisam ser referenciados, como o tipo de dispositivos de exibição de RA, como é feito o rastreamento de RA e como funcionam os robôs colaborativos. Também é investigado o tipo diferente de Interfaces de Usuário e como o conceito é aplicado à RA. Além disso, também é realizada uma pesquisa sobre a implementação de Sistemas Multi-utilizador por meio do uso de soluções de rede em um ambiente de RA.

Após assimilar os paradigmas investigados ao longo do relatório, propõe-se uma metodologia para alcançar o objetivo do projeto: com a ajuda de um dispositivo de Realidade Aumentada, um grupo de usuários deve ser capaz de pegar e controlar um modelo virtual da máquina simultaneamente. Os utilizadores também devem ter a possibilidade de visualizar as estatísticas da máquina e os seus processos devido à sobreposição de uma interface virtual ao campo de visão dos utilizadores. Os requisitos da arquitetura do sistema são, então, divididos em componentes menores para planejar a sua implementação, e, como o sistema precisa de suportar vários utilizadores em simultâneo, também é proposta a criação de um protocolo multi-utilizador por meio da aplicação e adaptação de uma biblioteca de rede de código aberto - Mirror. A arquitetura é planeada para ser capaz de gerar várias instâncias de servidor da aplicação num só endereço de IP aproveitando a multiplexação de portas e um servidor mestre que controla e monitora cada instância.

Em seguida, a implementação real do sistema é descrita em detalhes, assim como a incorporação de cada tecnologia no projeto. Os principais desafios que precisaram de ser superados para o sucesso do projeto e as suas respectivas soluções implementadas são expostas e explicadas. A integração de cada componente é elucidada em um método sequencial, decompondo o desenvolvimento do aplicativo em várias cenas. Por fim, o projeto é avaliado de forma a testar os seus limites e analisar potenciais áreas de melhoria, realizando assim múltiplos testes de stress no sistema do servidor. Também é tido em consideração o feedback dos utilizadores sobre a aplicação, a fim de verificar se está a funcionar conforme o esperado.

**Palavras-chave:** Realidade Aumentada, Interação Humano-Máquina, Interface de Utilizador, Robots Colaborativos, Colaboração Multiutilizador



# Acknowledgements

Este último ano foi longo e árduo, repleto de altos e baixos e de desafios pessoais e familiares. Sentia que sozinho nunca conseguiria alcançar os meus objetivos; contudo, o suporte dos que me são próximos incentivou-me imensamente, de forma a conseguir atingir os meus fins com sucesso.

À memória dos meus avós, que durante esta jornada partiram e deixaram uma saudade sempre crescente. À mulher mais amorosa e ternurenta, que me criou e fez de mim a pessoa que sou hoje – Júlia Martins; e ao homem mais astuto que alguma vez conheci, uma inspiração de pessoa que sempre procurava ultrapassar os seus próprios limites, e que indiretamente me levou a seguir a área de engenharia – João Gorra. Qualquer tipo de sucesso que venha a ter, tanto pessoal e profissional, deve-se a eles, e espero que lhes faça sempre justiça às pessoas que eles eram.

Aos meus pais e à minha irmã, por todo o suporte e amparo e pelos devidos conselhos que me davam, até quando não os queria ouvir. Por me deixarem desabafar, mesmo quando viviam conflitos pessoais consideravelmente piores do que os meus; e por estarem do meu lado quando finalmente termino uma grande etapa da minha vida, considerando todos os sustos e surpresas que apanhámos ao longo deste ano. Foram vocês que me permitiram chegar tão longe e não conseguiria agradecer o suficiente por sempre me guiarem na direção correta.

Gostaria de agradecer ao meu orientador, Marcelo Petry, pela sua constante preocupação e paciência. Mesmo quando via que o projeto não estava a progredir como previsto, sempre tentava redirecionar-me no sentido certo. Sob a sua supervisão fui capaz de melhorar as minhas capacidades, e graças à sua disponibilidade foi possível finalizar esta dissertação. Quero ainda agradecer ao INESC TEC e ao professor António Coelho pela oportunidade de poder trabalhar neste projeto, e ainda ao colega Gabriel Moura pelo seu bom sentido de humor e por me assistir na introdução à tecnologia de Realidade Aumentada e na utilização do HoloLens 2.

João Peixoto



# Abbreviations

6DOF	Six Degree of Freedom
API	Application Programming Interface
AR	Augmented Reality
CCU	Concurrent Users
CMC	Computer-Mediated Collaboration
CPU	Central Processing Unit
DNS	Domain Name System
DR	Duplicated Realities
FPS	Frames-per-Second
FTF	Face-to-Face
GPU	Graphics Processing Unit
HHD	Hand-Held Device
HLAPI	High-Level Application Programming Interface
HMD	Head-Mounted Device
HMI	Human-Machine Interaction
HPU	Holographic Processing Unit
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HUD	Heads-Up Display
IDE	Integrated Development Environment
IFR	International Federation of Robotics
IP	Internet Protocol Address
IPC	Inter-Process Communication
JSON	JavaScript Object Notation
LAN	Local Area Network
MR	Mixed Reality
MRTK	Mixed Reality Toolkit
MUVE	Multiuser Virtual Environment
NGO	Netcode for GameObjects
P2P	Peer-to-Peer
PHP	Hypertext Preprocessor
PUN	Photon Unity Networking
QR Code	Quick Response Code
REST	Representational State Transfer
ROS	Robot Operating System
RTT	Round-Trip Time
SHA	Secure Hash Algorithm
SLAM	Simultaneous Localization and Mapping

SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
UML	Unified Modeling Language
UNET	Unity Networking
UR	Universal Robot
URDF	Unified Robotics Description Format
UWP	Universal Windows Platform
UX	User Experience
VM	Virtual Machine
VPN	Virtual Private Network
VR	Virtual Reality
XML	Extensible Markup Language
XR	Extended Reality

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context and Motivation . . . . .	1
1.2	Objectives . . . . .	2
1.3	Problem Description . . . . .	2
1.4	Document Structure . . . . .	3
<b>2</b>	<b>Literature Review</b>	<b>5</b>
2.1	Augmented Reality Systems . . . . .	5
2.1.1	Extended Reality and the Reality-Virtuality Continuum . . . . .	5
2.1.2	AR and its Implementations . . . . .	6
2.1.3	Augmented Reality in Production Systems . . . . .	8
2.1.4	AR Display Devices . . . . .	10
2.1.5	AR Tracking . . . . .	13
2.2	Human Machine Interaction . . . . .	14
2.2.1	Human-Human Interaction . . . . .	14
2.2.2	HMI in Industry . . . . .	15
2.3	Multiuser Systems . . . . .	18
2.3.1	Multiuser Collaboration in an AR Environment . . . . .	19
<b>3</b>	<b>System Design</b>	<b>23</b>
3.1	System Requirements . . . . .	23
3.1.1	Multiuser Implementation Challenges . . . . .	24
3.1.2	Use-Cases and UML Analysis . . . . .	26
3.2	Technologies . . . . .	30
3.2.1	Microsoft HoloLens 2 . . . . .	30
3.2.2	Robot Operating System . . . . .	31
3.2.3	Unity . . . . .	32
3.2.4	Mirror Networking . . . . .	33
3.3	Proposed Architecture . . . . .	35
<b>4</b>	<b>System Implementation</b>	<b>39</b>
4.1	Environment Setup . . . . .	40
4.2	User Interface Design Choices . . . . .	42
4.3	Home Scene . . . . .	46
4.4	Session Scene . . . . .	49
4.5	Online Scene . . . . .	55
4.6	Application Deployment . . . . .	62

<b>5 System Evaluation</b>	<b>65</b>
5.1 Server-Client System Stress Test . . . . .	66
5.1.1 Testing Tools . . . . .	66
5.1.2 CPU and GPU Usage Tests . . . . .	66
5.1.3 Data Usage Tests . . . . .	69
5.1.4 Latency and FPS Tests . . . . .	74
5.2 Open Survey on UI/UX Demo . . . . .	75
<b>6 Conclusions and Future Work</b>	<b>81</b>
6.1 Conclusions . . . . .	81
6.2 Future Work . . . . .	82
<b>A Public UI Demo Test</b>	<b>85</b>
A.1 Public UI Demo Test Form . . . . .	85
A.2 Public UI Demo Test Form Responses . . . . .	94

# List of Figures

1.1 Simplified Use Case of the System . . . . .	3
2.1 The Reality-Virtuality Continuum (adapted from [5]) . . . . .	5
2.2 Example of multiple applications of AR respectively in Medicine [13] (Top image), for Military Training (Bottom-left image) [14], and in Education (Bottom-right image) [15] . . . . .	7
2.3 AR Interface for Order Picking Operations from [27]. The left image represents the highlighting of the order picking, and the right image is the visual navigation to the specific target bin . . . . .	9
2.4 Architecture for an optical see-through display device (top image) and a video see-through display device (bottom image) (adapted from [33]) . . . . .	11
2.5 Marker-based architecture for AR tracking and rendering . . . . .	13
2.6 Markerless-based architecture for AR tracking and rendering . . . . .	13
2.7 Different synergy types between a human and a collaborative robot (adapted from [48]) . . . . .	16
2.8 INESC TEC's Programming by Demonstration Software (from [63]) . . . . .	21
2.9 Generating a Virtual Model by Tracking the Interaction with Kinetic Sand in a Duplicated Reality Environment (from [66]) . . . . .	22
3.1 Login Sequential Use-Case Diagram . . . . .	26
3.2 Create and Join Session Sequential Use-Case Diagram . . . . .	28
3.3 Robot Control Sequential Use-Case Diagram . . . . .	29
3.4 Example of an AR environment and the interaction between human and machine via gesture-based inputs on a holographic 3D model of the robot [67] . . . . .	30
3.5 ROS architecture for a subscription-based system . . . . .	31
3.6 Overview of the System's Architecture . . . . .	36
3.7 System's Architecture with One Instance of a Session . . . . .	37
4.1 Simplified App Flow UML . . . . .	40
4.2 XAMPP Control Panel . . . . .	41
4.3 Low-Fidelity Prototype of Log In UI . . . . .	43
4.4 First and Last UI Iteration Cycle . . . . .	44
4.5 Side-View of the AR Interface of a Scrollable List . . . . .	45
4.6 Simplified Home Scene Flow UML . . . . .	46
4.7 Authority Hierarchy . . . . .	47
4.8 Login and Registration Website . . . . .	47
4.9 Users Table SQL . . . . .	48
4.10 Login AR Interface . . . . .	49
4.11 Simplified Session Scene Flow UML . . . . .	50

4.12 Custom Network Manager and KCP Transport Components . . . . .	51
4.13 Creation of two Sessions . . . . .	52
4.14 Player joins the Session with Port 10004 . . . . .	53
4.15 Database Entity Relationship Diagram . . . . .	53
4.16 Create Session AR Interface . . . . .	54
4.17 List and Join Sessions AR Interface . . . . .	55
4.18 ROS Tech Demo for UR10 . . . . .	56
4.19 Users' ROS Commands are routed through the Session Server . . . . .	58
4.20 Robot's Programming Target . . . . .	58
4.21 Robot's Workspace . . . . .	59
4.22 Player Prefab above Users 1, 3 and 4 Heads . . . . .	60
4.23 Robot Model appearing over QR Marker . . . . .	60
4.24 Independent Session UI . . . . .	61
4.25 Login AR UI on HoloLens 2 . . . . .	63
4.26 Session Creation AR UI on HoloLens 2 . . . . .	63
4.27 Joining Session AR UI on HoloLens 2 . . . . .	63
4.28 Manipulable Robot Synchronizing both on Client and Server's ends . . . . .	64
4.29 Manipulable Robot during Session and User Prefabs above Participant's Heads . . . . .	64
4.30 Session Control AR UI on HoloLens 2 . . . . .	64
5.1 CPU Usage per Session Instance . . . . .	67
5.2 GPU Usage per Session Instance . . . . .	67
5.3 CPU Usage per Client Instance in a Session . . . . .	68
5.4 GPU Usage per Client Instance in a Session . . . . .	69
5.5 Number of Packets Sent by the Session per Second . . . . .	70
5.6 Packets Size Sent by the Session per Second . . . . .	70
5.7 Number of Packets Received by the Client in the Session per Second . . . . .	71
5.8 Packets Size Received by the Client in the Session per Second . . . . .	71
5.9 Number of Packets Received by the Session per Second . . . . .	72
5.10 Packets Size Received by the Session per Second . . . . .	72
5.11 Number of Packets Sent by the Client in the Session per Second . . . . .	73
5.12 Packets Size Sent by the Client in the Session per Second . . . . .	73
5.13 Client RTT per Client Instance in a Session . . . . .	74
5.14 Form Testers Age . . . . .	77
5.15 Form Testers Gender . . . . .	77
5.16 Form Testers Opinion on Login Menu UI . . . . .	77
5.17 Form Testers Opinion on Session Creation Menu UI . . . . .	78
5.18 Form Testers Opinion on Joining Session Menu UI . . . . .	78
5.19 Form Testers Opinion on Side Menu UI Switching . . . . .	78
5.20 Form Testers Opinion on Robot Manipulation . . . . .	79
5.21 Form Testers Opinion on Online Session Menu UI . . . . .	79
5.22 Form Testers Final Opinion on the UI . . . . .	80
5.23 Form Testers Opinion on the Difficulty of the UI . . . . .	80

# List of Tables

2.1	Summarized comparison between the different type of AR display devices (adapted from [6]) . . . . .	12
3.1	Summarized comparison between the different netcodes (adapted from [72]) . . . . .	35
4.1	System's Port Allocation . . . . .	52
5.1	Form Testers Occupation . . . . .	77
5.2	Form Testers Opinion on UI . . . . .	80



# Chapter 1

## Introduction

### 1.1 Context and Motivation

The advancement of Industry 4.0 in modern companies and factories empowers the possibility of shorter production cycles through the adoption and usage of more efficient and technological equipment [1]. Alongside the development and implementation of this industrial stage, it is also recognized an increase in complexity in the production systems needed for both regulation and operation [1], [2]. Thus, the need for new and simpler methods of interaction between the operators and the machinery with the purpose of not only creating intuitive mechanisms but also immersing the user in the experience arose the possibility of integrating Augmented Reality (AR) with the production systems. Applying this concept enables a way of overcoming extensive and fallible system mechanisms by meeting a halfway point between reality and technology [3].

Another essential aspect that characterizes Industry 4.0 is the need for a personalized production where the customer can order unique, customized products, therefore the idea of a serialized and linear mass production system is becoming obsolete. Alongside with the high demand for product quality and the industry's heavy, competitive environment, the need for a more flexible and distributed industrial operation emerges [4]. In order to achieve this, the machinery and the industrialized procedures become even more convoluted which, consequently, demand a higher level of expertise from the operators. This may result in a more intricate and user-hostile interface that can be sorted out by applying an adjustable interface combined with AR technology to the system.

Despite this improved system still being fully automatized, since it relies on the human operator's input for certain aspects of the task, the workspace must not only remain safe for both elements of the collaborative session but should also be designed to achieve an ideal and optimized productivity level. The result is a new set of technologies and paradigms applied to the industrial system, such as collaborative robots and Human-Machine Interaction (HMI). By applying and combining these concepts it is possible to attain an efficient and dynamic collaboration between the operator and the machine that results in an overall optimized system [3].

## 1.2 Objectives

This dissertation proposes a solution for some of the problems brought up by Industry 4.0 and the ever-increasing complexity of the tasks by integrating Augmented Reality into the User Interface, and through the promotion of user collaboration using a multiuser system. By expanding on the Human-Machine Interaction concept, it is possible to achieve an intuitive and enhanced interaction between humans in an online session in order to solve a task of high-complexity in tandem. These technologies' state-of-the-art were investigated and developed in order to plan their future implementation and research how they could solve some of the industry's issues. By taking advantage of the AR system's characteristics, the operator should be capable of effortlessly manipulate the collaborative robot simply through gesture-based controls. The interface should be displayed to the operator through a Head-Mounted Device, the Microsoft HoloLens 2.

In addition, the project also integrates a framework capable of supporting multiple simultaneous users in the same AR experience through the adoption of network libraries. The AR-driven system should be apt to connect to a production system's database and present relevant information related to a process state or equipment details. An architecture capable of supporting this framework is designed for later implementation, testing and evaluation. The referred system could severely boost efficiency; improve safety and control protocols or fix complications by overseeing the systems' statistics and previous history; it could also enable for rapid and flexible training of staff due to the intuitive interface and multiple/simultaneous user interaction, and an overall long-term profit for the company.

## 1.3 Problem Description

With the main goal of taking Human-Machine Interaction to the next level in Industry 4.0, the implementation of AR in machinery could prove itself to be especially useful to the user experience and learning curve of the workers. Through the use of an AR device - in this case, the Microsoft HoloLens 2 - an operator should be capable of visualizing the robot being used in the Human-Machine collaborative session, with the possibility of showing specific statistics and data about the processes currently being performed, and also, interact with a Digital Twin of the collaborative robot in order to operate its real counterpart. A very simplified use case diagram of the system can be analyzed in Fig. 1.1 in order to understand better what the system should be capable of after implementation and testing.

However, the focus of the project is the multiplayer feature and the possibility of mutual co-operation and collaboration among teams and workers. Since this project by itself is already very visionary, with plenty of high-end technologies, and abundant potential for future additions, - such as image recognition, Artificial Intelligence, cloud computing, IoT, assistive and expansive operational technology even for heavy manufacturing processes, and compatibility with other AR devices, such as the end-client's personal mobile phone or Personal Digital Assistant (PDA) - the

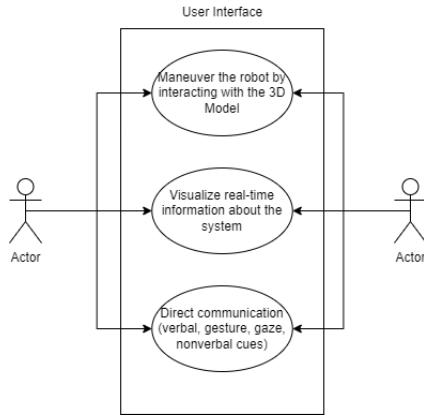


Figure 1.1: Simplified Use Case of the System

result of this dissertation shall act as a supposed base main framework for all these possible technological advancements with the Client-Server model already implemented, with full access to a database, a master server and full connection to the collaborative robot.

Although there are plenty of challenges with the actual design of the system. For instance, there is no standardized process for UI design and UX optimization in AR, since it's still a rather early concept that is very recently being further explored. Likewise, multiuser experiences in an AR environment can be particularly challenging due to the absence of consistent network implementation processes or dedicated packages or libraries for the technology in specific - unlike Virtual Reality which is able to use modern network libraries to their maximum potential, since VR disconnects the user from the real world and is fully virtual, like a 3D game in first person. The lack of complete guidelines in terms of designing online experiences for AR may impose the need to adapt existing technologies to suit the project's goals.

## 1.4 Document Structure

This document is divided into four main chapters, excluding the present one (Chapter 1). The structure adopted is as follows:

- In Chapter 2, it is made an extensive review of the literature and implementations related to Augmented Reality, Human-Machine Interaction and User Interfaces. It is briefly introduced the Reality-Virtuality Continuum, the evolution of AR systems applications throughout history and some recent and relevant implementations on industrial environments. Still related to AR, it is examined the different type of AR devices and how the tracking is carried out. Additionally, HMI is also evaluated considering the account of how robot collaboration is achieved in the industry. This section also develops on multiuser systems, how they are planned the requirements for their implementation, also some network models are analyzed. Finally, some pertinent works related to multiuser collaboration through AR are also examined.

- In Chapter 3, it is described the collaborative framework, the system's architecture, and all the technologies expected to be included in the implementation of the project. It expands on the requirements set for the project, by not only explaining the prerequisites set initially as the project's goals but also considering some of the components analyzed during research in Chapter 2 that would allow the successful implementation of the system. Then the multiuser implementation challenges are studied and how the actual development of the project should solve these issues. In order to understand better the system, some use-cases and UML diagrams were studied, which should be helpful for the later designing the system's architecture. Every major technology incorporated into the project is systematically reviewed in a more thorough way, and how they should interact among themselves. Finally, after considering all these aspects, the system's definite architecture is proposed - an architecture that works in theory while still being flexible enough to accommodate for future development and with scalability in mind.
- Chapter 4 focuses on the actual implementation of the system, and overall, committing to the architecture proposed. First, the main development challenges between the transition from theory to practice are analyzed, to check if the project is ready for this stage; then the application development is decomposed into multiple components in order to explain in an organized and sequential way the incorporation of the technologies into the application. The first component analyzed is the setup of the computer environment since it is the foundation for the development of the system; then the thought process and the methodology behind the planning and design of the User Interface is also examined. Then, every major scene from the application is explained in detail: first, the login and registration mechanism, then the creation and joining of online sessions, and finally the actual online networked session. The main aspects mentioned in these scenes regard the database usage, the creation of the master server, the network programming, all the back-end programming and the UI conception. In the end, the actual employment of the system on the target AR device - the HoloLens 2 - is performed, and all the debugging and testing process is explained.
- The Chapter 5 aims at evaluating and testing the system created to find critical points of failure and the project's limits. Multiple tests were performed on the server-based system, such as CPU and GPU usage, latency and bandwidth management, and the application FPS utilizing a wide range of tools. These tests are relevant in order to locate areas that may be in need of improvement. In addition, the UI was publicly deployed on a web-based application (although, without the multiuser experience) so that it is possible to assess the overall user experience and learn from the feedback of users on what should be improved in a future iteration.
- The final Chapter 6 presents a summary of the project and if its main goals were achieved. Conclusions are drawn according to the research previously made, and if the work that was planned for was successfully implemented. It is also taken into consideration future additions to the project, considering the system's scalability and visionary outlook.

# Chapter 2

## Literature Review

### 2.1 Augmented Reality Systems

Modern investigation on Augmented Reality-driven (AR) devices is rapidly growing, therefore the need for a thorough research is essential to discern the best approach to the problem in hand. Thus, this section is dedicated to exposing some of the topics surrounding the theme of this dissertation. It will be discussed essential concepts, their problems and corresponding solutions, and some of the latest or most relevant research made on them.

Initially, it is examined how the term AR came to be; then an extensive analysis on the modern applications of AR and some of the history behind it focusing mainly on its industrial potential aspects. Next, it is presented all the different kinds of AR devices and their main assets which is followed by how the AR tracking technology is performed.

#### 2.1.1 Extended Reality and the Reality-Virtuality Continuum

In order to label the combination of real and virtual environments, the reality-virtuality continuum was defined by Milgram in 1995 [5] as a scale that ranges from exclusively real to exclusively virtual. The result is a measure that subsumes every possible combination of both real and virtual elements. This made it simpler to characterize applications that would occasionally fall in-between concepts, considering the terms and categories akin to the reality-virtuality compositions were still very broad and inconsistent as well.



Figure 2.1: The Reality-Virtuality Continuum (adapted from [5])

The figure 2.1 details the scale and its corresponding designations. The left extreme is the real environment as we know it, and the respective utmost point is the complete immersion in a digital environment. The real environment is composed of only corporal and physical objects, in opposition to the Virtual Reality (VR) wherein there's just computer-generated elements [6]. Amidst these notions is the Mixed Reality (MR) which is composed of both real and virtual objects, presented together in a single display. Mixed Reality is also divided into two major categories: the Augmented Reality (AR), where the real world is augmented through the use of virtual objects; and the Augmented Virtuality (AV), which contrariwise to AR, augments the virtual world by integrating real objects.

### 2.1.2 AR and its Implementations

AR is a technology that enhances the real-world environment by superimposing virtual elements. It enables for an interactive experience for the user through sensory stimuli – such as visual, auditory, olfactory, and others – via computer-generated information. This digital information is delivered in such a way that in the user's perspective it looks like it is part of the real world, which results in a synergistic connection for both the user and the computer [7].

Historically, one of the first approaches to AR was through the Sensorama in order to provide a more realistic cinematic experience in the late 1950s [8]. In the following years, AR was not really considered a priority considering the low computational power of the machines. In 1968, Sutherland developed the first Head-Mounted Display (HMD), which changed the course of development of AR due to its innovation and potential in the modernization of task execution. This device was mechanically tracked through accessories artificially placed in the ceiling, named as “Sword of Damocles”, and allowed the user to see-through the device and visualize the room combined with 3D computer-generated graphics.

The first use of the term “Augmented Reality” was made by Caudell, a former Boeing researcher, in 1990 [9] and two years later Caudell and Mizell developed an early AR prototype capable of projecting assembly blueprints onto a surface. With this implementation, it was concluded that AR technology could “enable cost reduction and efficiency improvements in many of the human-involved operations” [9].

With the evolution of Hand-Held Devices (HHD) and the progressive standardization of devices like the smartphone or the tablet in society, new opportunities for the development of AR appeared. In 2013, Google promoted their new AR device, the Google Glass. It was an HMD system that, through the interaction between the user and a voice assistant, enabled for a hands-free experience replication of a smartphone. The user could make phone calls, text and navigate the Internet. Later on, in 2015, Microsoft introduced the HoloLens which was an especially powerful system that enabled the user to interact with the virtual environment via gestures, voice and gaze. At the time, due to the system being easy-to-use, its affordability in industrial means, and its overall use potential, paved a way to the rapid advancement of AR technologies and its modern application in Industry 4.0 [10].

Nowadays, the use of AR has been extensively explored and applied to several fields, like in education, product design, medicine and many other cases [11]. According to a survey by van Krevelen et al. [7], it is expected that in the near future, society completely adopts AR technologies in order to enhance and accelerate otherwise common activities, going from its prevailing usage in manufacturing into homes, hospitals and schools, for example. Some of these areas already started applying AR into their tasks.

Thad Starner et al [12], predicted this trend, and the possibility of integrating AR into people's personal life in pursuance of convenience and comfort. In this research, through the means of biosensors, displays, and even smart clothing, the person could store relevant data fluidly at any time and have a real-time access to it. The implementation of these systems allowed for other impressive applications, like an informative adaptive Heads-Up Display overlay giving instant information on what the user was visualizing, face recognition, and an assistant for visually impaired people that would apply a "hyper-fisheye" filter to magnify the imagery. These kinds of applications have a straightforward purpose: to facilitate the user's lifestyle and improve their quality of life.

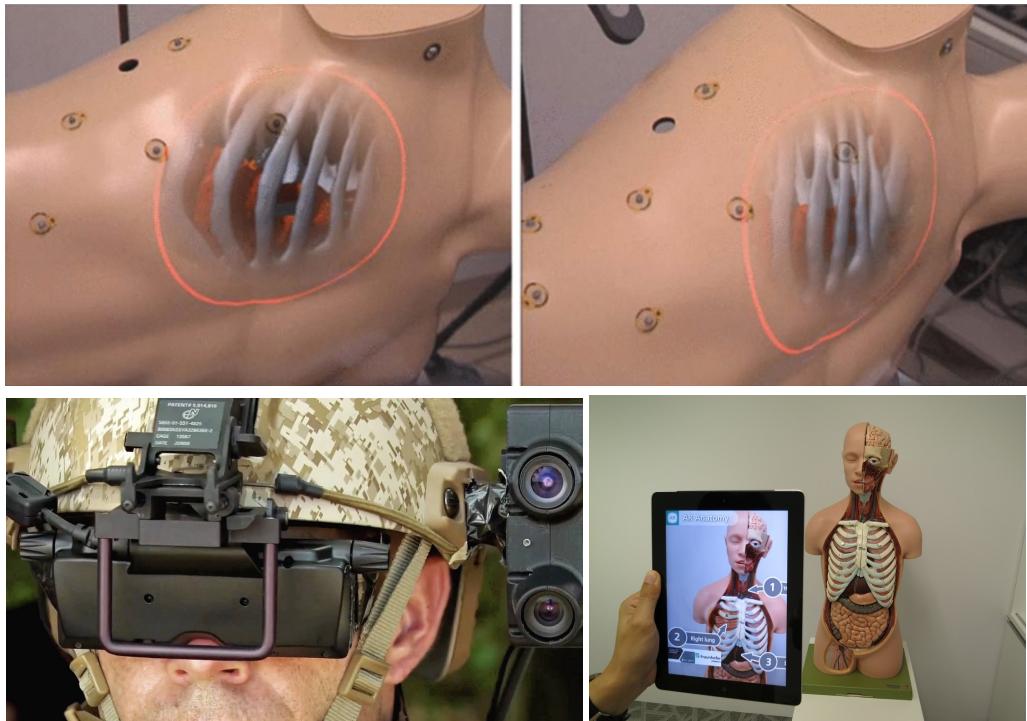


Figure 2.2: Example of multiple applications of AR respectively in Medicine [13] (Top image), for Military Training (Bottom-left image) [14], and in Education (Bottom-right image) [15]

Currently, AR systems are also applied in tourism areas, such as through the touring of Virtual Heritage (VH) environments [16], [17], or by overlaying the vision of the visitor with relevant information on the cultural piece [18]. Virtual Heritage is described as the fusion of Mixed Reality technology with cultural heritage content, and its main goal is to preserve digitally these historic

objects that would otherwise decay, and also, to explore them remotely in a close-to-realistic way. In Basel, it is possible to take an AR tour that, through direct access to a database dedicated to cultural purposes, it displays information about the city and its outskirts regarding relevant sites, museums, current events, and much more [19].

The advancement in modern medicine goes hand-in-hand with the development of technology, therefore, it is expected that this field should also be able to avail from AR benefits. Jeffrey Shuhaiber [20] analyzed this position minutely and came to several conclusions: even though the use of AR in surgeries is not completely established due to the many unique variables on different human bodies, it is however very advantageous as an assistant to the doctor. By providing relevant points, lines and planes as landmarks on the human body, the medic can perform surgery taking in consideration the optimized points of incision or perforation. It also superimposes the surgeon's view with relevant data from Computed Tomography scans or Magnetic Resonance Imaging results, for example.

Still, there are plenty of other compelling areas where the application of AR benefits the user. The Handbook of Augmented Reality [8] and many other sources [21] explore further modern employment of the system in topics like Nano Manipulation, Psychology, Education, Environmental Planning and Military. The figure 2.2 references visually some instances of these applications.

### 2.1.3 Augmented Reality in Production Systems

In industrial environments, an AR-driven system combined with HMI "is applied in order to raise efficiency, quality, and process safety" [22]. A review on the modern applications of AR in industrial environments [23] concluded that some of the most relevant topics researched focused on the development of these fields:

- Assembly
- Maintenance
- Manufacturing
- Design or Prototyping
- Order Picking

Nowadays, assembly and maintenance operations are commonly set up for computer or tablet displays, however, as stated by Gabriel Evans et al [10], these impose very strict limitations that ultimately disrupt the process. In the case of computer displays, they are more often than not planted in a fixed location. This contributes for a very rigid, harsh interaction with considerable time losses due to the necessity of the operator having to move away from their working position and refocus on a different visual target, possibly out of the field of view. Tablets fixed most of these issues related to the physical mobility of the user, however they still pose a critical issue: the operator still needs to divert their hands and attention [10]. AR Head-Mounted Devices (HMD)

pose as a solution to this issue by allowing users to visualize real-time instructions while still having complete freedom to use their hands. This way, the flow of the assembly or maintenance operation is not interrupted or temporarily halted which contributes to a higher level of efficiency.

AR can also help operators and engineers by providing digital instructions for manufacturing processes [24], [25]. By overlaying the operator's field of view of the machinery it is possible to give direct instructions on the working area directed to certain components of the machine. It can also simulate tasks, give an immediate unit analysis and, since it is all performed virtually, it can give a fast access to past analysis for a history comparison. It also works as a real-time smart assistant by providing guidance to the user via real-time alerts, notifications and other relevant information.

Additionally, AR can be used in improving data management practices and as a visualization tool for product design or prototyping. Presenting the three-dimensional model of a product eases the understanding of its spatial context and its integration with the rest of the system; it also results in a simplified product comparison of possible alternative solutions and a streamlined testing simulation. Y. Shen e al. [26] suggests a framework oriented to product prototyping with the capacity of local multiuser collaboration based in an AR environment. The main conclusions drawn about this kind of employment of AR in industrial means were that it severely reduced the redesign iteration and costs.

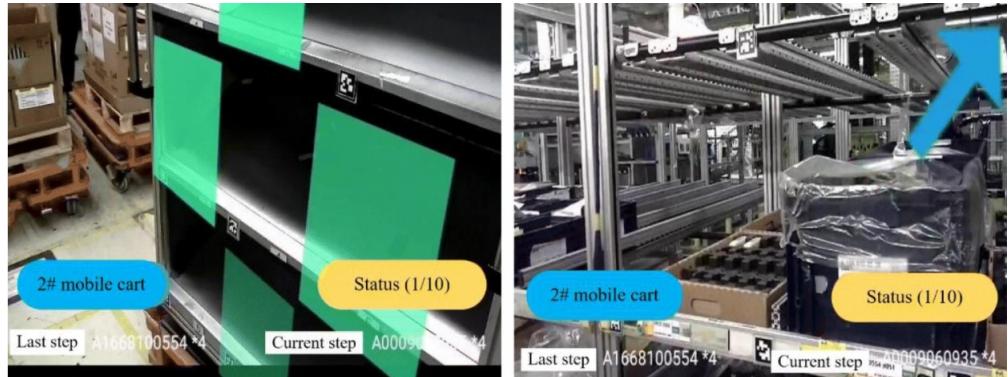


Figure 2.3: AR Interface for Order Picking Operations from [27]. The left image represents the highlighting of the order picking, and the right image is the visual navigation to the specific target bin

Order picking is a very important process step in logistics, but, as reported by Wei Fang and Zewu An [27], having to rely on a paper-based listing of the product's identifier, location and amount sets up for a stress-inducing and fatiguing experience for the pickers, especially considering the cluttered and piled characteristics of a warehouse and the high repeatability of these tasks. The system proposed by Wei Fang and Zewu An [27] is based on the integration of AR through HMD devices. Globally mapping a marker-based warehouse floor achieves efficient navigation for the picker and, due to the capability of superimposing the information on the users field of view, the operators can perform picking operations without unnecessary mental efforts. The application of AR in picking processes via HMD devices, according to research [28], allows

for untrained pickers to complete orders 37% faster when compared to the common paper-based listing methods.

There is a vast number of modern research documents verifying that an implementation of AR-based technologies on industrial settings result in compelling outcomes, such as [9], [29], [30], and as seen in this section. An important detail that should also be mentioned is that AR systems are easily scalable, allowing for ease of future implementation and a better return on investment.

#### 2.1.4 AR Display Devices

Most modern AR systems can be broken down into multiple underlying and essential constituents, such as the infrastructure tracking unit, the processing unit and the visual unit. The infrastructure unit is responsible for reading the real world's data through the use of sensors, and for the tracking and positioning of the user in the environment around them. The visual unit is capable of capturing the real world for computer vision and image processing in certain devices and then also show the user the output created by the processing unit. Lastly, the processing unit accounts for all the complex data computation from the sensors, tracking and visualization and then adds the virtual content which results in an intricate mix of both realities. These devices used for AR visualization and interaction can be categorized primarily in four broad classes [31]: Heads-Up Displays, Spatial Displays, Head-Mounted Displays and Hand-Held Devices.

Heads-Up Displays (HUDs), initially developed for aircrafts, focuses in providing an overlay of additional information to the user which increases situation awareness since it reduces the need of looking away from the display. This device is mainly composed of three constituents: the display screen or surface, a projector and the computer. An example of a hands-free HUD system was researched by Anup Doshi et al. [32] in 2008 with the goal of implementing a driver assistant that provides real-time alerts of critical situations. While avoiding visually cluttering the field of view of the driver, it would track the head movement of the person as means of interaction and provide relevant information about the road on the car's windshield through laser reflection.

Head-Mounted Display (HMD) is a wearable device that superimposes information alongside to what the user is currently seeing. Since it produces a hands-free experience, it can feel the most ergonomic and natural, however, it can also cause eyestrain when used for long periods of time. It is divided in two categories: optical see-through and video see-through. In the former the user visualizes reality directly and in the latter the user actually sees a video of the environment in almost real-time captured by one or two cameras. The greatest disadvantage of video see-through systems is that the video resolution and slight delay end up damaging the complete immersion of the user, although its environments simulated offer a higher quality adjustability and reliability when compared to optical see-through devices. The baseline architecture and components of both the optical and video see-through display devices can be observed in Figure 2.4.

The Hand-Held Device (HHD) is the most accessible technology to the common person since a simple smartphone is capable of creating an AR experience. All handheld solutions to date are categorized as a video see-through device where the the real environment is recorded through the device's built-in camera, combined with the virtual objects and then displayed to the user. Due

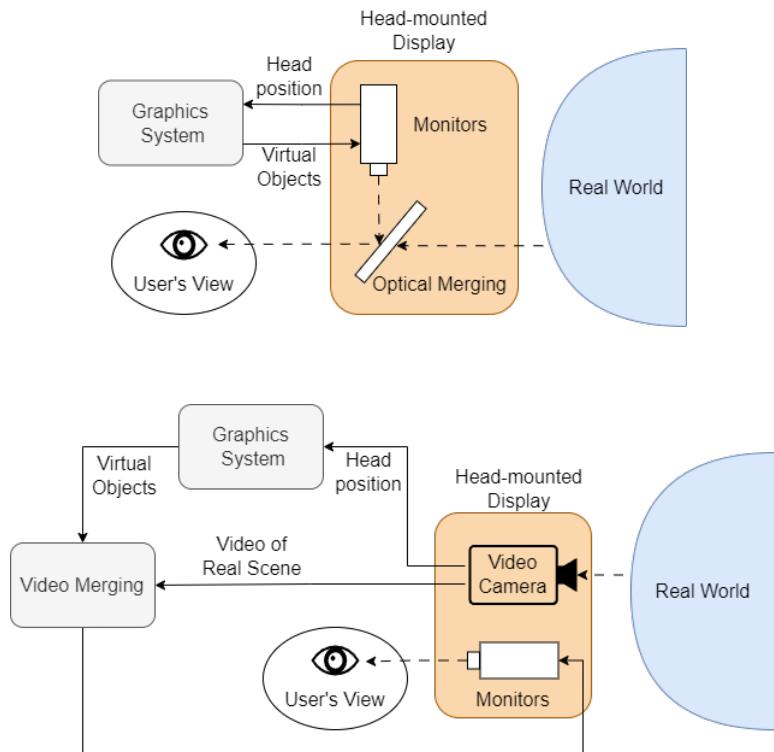


Figure 2.4: Architecture for an optical see-through display device (top image) and a video see-through display device (bottom image) (adapted from [33])

to their portability and convenience, this type of devices is being one of the main focus for AR development nowadays. However, HHDs also heavily handicap the user by requiring them to hold the device during operation and by demanding the operator to refocus their attention from the task. Mobile AR systems also face other issues, for instance, if the GPS system is not accurate enough, the virtual objects can be misplaced due to the lack of precision, and usually these devices have very strict hardware limitations that may be harmful for the experience when processing the virtual environment.

Holographic or spatial displays is a technology capable of creating 3D models without the need for the user to carry additional gear. It works by diffracting the light of a laser in a specific display area and can be composed of multiple projectors [34]. Even though this kind of devices is physically strict considering that the virtual image is displayed in a static location, it is ideal for collaborative tasks among multiple operators as the virtual environment is shown to everybody simultaneously, without the need of additional gear. Nonetheless, this type of AR display still has some restraints to work, for instance, in order to create the illusion of a hologram it needs to not only track the user and the environment, but it also requires to know about the shape of the surfaces where the rendering of the virtual objects will occur. In order to summarize the main pros and cons of every AR display device researched, it is possible to use Table 2.1 for a straightforward comparison.

Table 2.1: Summarized comparison between the different type of AR display devices (adapted from [6])

Display type	Optical		HMD		HHD		Spatial display	
	Video	Projective	Video	Projective	Video	Projective	Projective	Projective
Mobility	+	+	+	+	+	-	-	-
Interaction	+	+	+	+	+	-	-	-
Outdoor utility	±	±	±	+	+	-	-	-
Multiuser	+	+	+	+	+	Limited	Limited	Limited
Brightness	-	±	±	±	Limited	±	±	±
Contrast	-	±	±	+	Limited	±	±	±
Resolution	±	Limited	Limited	Limited	Limited	+	+	+
Field of view								
Pros	Current dominance		Mass-market and accessible		Cheap and ergonomic		Image clipping	
Cons	Tracking difficulties		Delay and resolution		Limited hardware		Image clipping	

### 2.1.5 AR Tracking

In order to track the environment and the user's relative position, the AR device has mainly two options, to use a physical marker – like a QR code – or to use a computation-heavy algorithm for mapping the entire environment and finding important reference points. Marker-based techniques require the use of tags artificially placed in a real environment, and since its detection is straightforward it ends up requiring less computational power [35]. In order to identify the markers, the device resorts to computer vision processes and algorithms that assist in not only detecting unique markers but also in calculating the relative position and orientation of the device to the marker. The architecture behind the processing of a marker-based algorithm can be analyzed in Fig. 2.5.

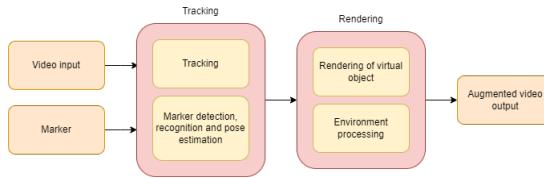


Figure 2.5: Marker-based architecture for AR tracking and rendering

Contrarily, markerless techniques rely on detection through computer vision methods like edge and corner detections and texture identification. In computer vision, most of the techniques can be grouped into two main categories: feature-based and model-based [36]. While the former relies on forming a combination of lower-level features by connecting 2D image features to their 3D world frame coordinates, the latter tries to fit models tracked to lower-level distinguishable features. Ultimately combining both approaches, by scanning the surrounding real environment and recognizing prominent features in it, it is capable of integrating virtuality using the computer-generated flags extracted as guides for the device's distance and orientation. This results in a very flexible and versatile tracking with high processing dependence since this method usually leans on database-heavy operations for the identification of models and their textures. It is possible to examine the architecture for the markerless algorithm in Fig. 2.6.

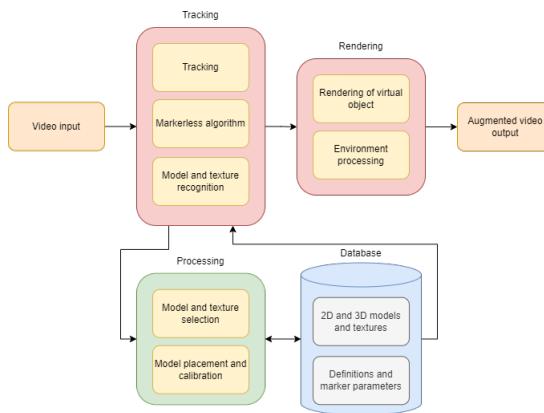


Figure 2.6: Markerless-based architecture for AR tracking and rendering

Since AR environments rely mainly on the user for interaction with the digital objects, the tracking system must deliver high accuracy, even at long distances, and also an expanded selection of possible valid inputs [37]. Six degree of freedom (6DOF) is a concept that describes the ability of an actor to move without restrictions in the three-dimensional space [38]. In AR, 6DOF tracking is the vital mechanism responsible for the sense of realism and immersion for the user by capturing their position and movement. This great need for a sharp precision generated an ever-evolving assortment of technologies: in early prototypes, the positioning tracking had to be made under very specific conditions, and as such, it would be required an indoor setting where the brightness and temperature could be manipulated, and where the system could resort to the usage of artificially placed sensors in the room. The lack of adaptability of this technique made others, more versatile, methods surface. Due to the advancement of sensors and their proper capacity, the application of inertial sensors, like gyroscopes and accelerometers, combined with 3D cameras, localization algorithms, and marker or markerless tracking techniques result in a very accurate tracking of the 6DOF, and therefore, a more fluent immersion for the user.

## 2.2 Human Machine Interaction

Due to the focus in large scale manufacturing, industries need high levels of automation, however, considering some tasks high-complexity and expensive automation, certain systems should be designed in a way that complements the operator. This paradigm known as Human-Machine Interaction (HMI) happens when there's a combination of a machine's capability and a human's insight, resulting in an overall improvement on the task execution's efficiency. Before analyzing how the interaction between a human and a machine is made, it is essential to comprehend how people socially relate between themselves since these factors also apply to Human-Machine interaction [39].

Human-machine interaction covers multiple topics, like human-computer interaction, human-robot interaction, artificial intelligence, and robotics [40]. Combining all these technological fields in an industrial environment allow for capable, dynamic teamwork between the robot and the operator, especially when considering the concept of collaborative robots designed to enhance the interaction between both parties. Furthermore, there are different kinds of synergy between the human and the machine and each task's procedure should be deliberately planned for these collaboration methods to optimize the system's efficiency. Also, the application of these robotic concepts also imply specific safety protocols to be implemented in the workspace in order to avoid harm on both ends.

### 2.2.1 Human-Human Interaction

Human communication is considered to be multi-modal since it relies on speech, gesture and gaze [41], which vastly benefits face-to-face collaboration. In addition to these methods, people also depend on other non-verbal cues, like posture, or a combination of modes, for example, pointing at an object. Interactions with the real world also proves to be valuable in social situations. So, a

robot implemented to collaborate with a human should be aware of its environment while being capable of using speech and gesture as guidance and should also be able to capture and emulate nonverbal cues.

Moreover, if instead of relying on face-to-face collaboration a person had to cooperate remotely it should lead to an extensive loss in the richness of this data. For instance, by considering the case of two persons cooperating through a phone call, there is no visual or environmental data to process, therefore contributing to a substandard experience. In this case, the person might need to explain situations through the use of longer sentences which increases the time needed to transmit an idea and an overall slower communication between both. A machine should be capable to not only communicating clearly with a human through verbal, visual and environmental channels, but should also have the ability to compensate in case one of these channels is not present.

As seen before, the main way people interact socially is either through linguistic or visual means however in this pretty straightforward method of communication there is also plenty of “hidden”, subjective information. Depending on the person’s motivation, attitude, perception and interpersonal skills, there is a transmission of extra data through other nonverbal and micro-expressions [41]. Additionally, there’s a social influence that results in non-conscious mimicry of postures, mannerism and facial expressions – the chameleon effect [42] – which leads to irregular and hard-to-predict data for a machine. This data interpretation can be achieved in mainly two ways: through the implementation of computationally heavy Artificial Intelligence algorithms that analyze the user’s visual and nonverbal cues with a high efficacy level, or by simply making the user aware of these machine faults in order to transmit the data as eloquently and directly as possible.

### 2.2.2 HMI in Industry

At par with the growing intricacy of machines and robots, their autonomy and capacity also evolve, becoming even less dependent of people. However, a fully automatized system is sometimes impossible or extremely difficult as a consequence of highly complex business models or the occasional need for a customized item from a mass manufactured product line [43]. And as pointed out by Inagaki et al. [44], by varying the levels of autonomy from both the user and the machine it is possible to optimize the interaction by relying on the different strengths of both parties. The human is considered to have the problem-solving skills and the business perception, while the machine has the power, deftness and velocity, though, modern applications of machinery are capable of quickly learning these skills from the human and apply them effectively [45]. These applications rely on the implementation of Artificial Intelligence systems, and even though they can be fully autonomous, the system is also capable of detecting when the human input should be pertinent to the quality of the task.

Yet, most industrial machines are designed with its task execution efficiency in mind and are not supposed to have direct contact with the human, in fact, the machinery is usually isolated from the operator zone to avoid potential safety hazards. To facilitate the interaction between human and machine, the concept of a collaborative robot, also known as robotic assistant or *cobot*, was

elaborated. Even though these robots do not need to have a distinct appearance from the standard industrial machines, they need to follow an extra set of technical specifications and regulations to safely support human interaction [46]. The cobots are meant to complement the machinery system and not substitute it, as they lead to plenty of benefits by their inclusion in the production system according to Ales Vysocky and Petr Novak [47]:

- Provide higher competitiveness between companies, especially when compared to countries with cheap labor
- Reduce the need for post processing and quality control due to the robot's accuracy and continuous operation
- Accelerate operation and increase productivity by adjusting to special conditions
- Limit repetitive and tedious work by relieving the operators of stressing tasks, which also reduces the chances of an occupational disease and injuries
- Establish a safe environment, where dangerous situations should only occur due to the circumvention of safety regulations

With the evolution of the cobots, the workspace shared between the human and the machine keeps on expanding, which means that, in the future, this space will be eventually undivided [47]. Depending on the task's nature and intricacy, there are multiple types of HMI available, as seen on the Figure 2.9, where the circle represents the workspace, and the square is the task or product's part that is being worked on.

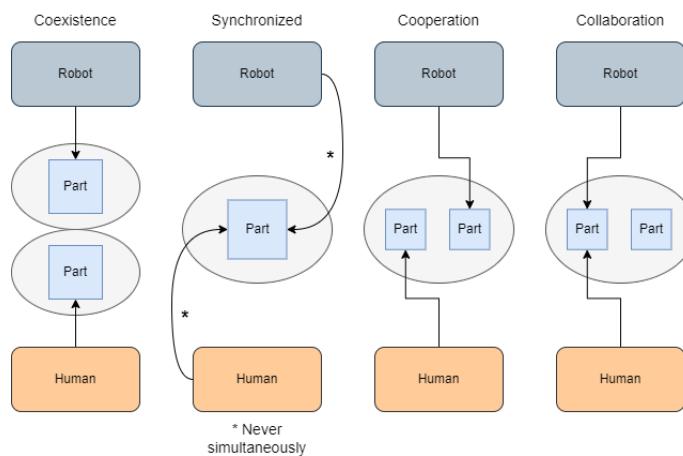


Figure 2.7: Different synergy types between a human and a collaborative robot (adapted from [48])

There are mainly four kinds of HMI: coexistence, synchronized, collaboration and cooperation [48]. Coexistence is the simplest and has the lowest requirements for implementation, where the robot and the human merely operate in the same environment but do not interact with each other at all – they work individually on their own task. Synchronized interaction conveys that both the

human and the robot will eventually work on the same workspace and on the same task, but never simultaneously.

Cooperation and collaboration, even though these terms share their general meaning, in HMI they have different interpretations [49]. Collaboration is a more restrictive and precise meaning whereas cooperation is more of an umbrella term for any interaction. To collaborate in a certain task is to partake in a combined and shared cooperative activity [49], [50], thus, both terms should be differentiated in the types of HMI. So, in the collaboration approach, the human and the robot combine strengths and work in tandem for the same objective by interacting directly on their mutual task. On the other hand, in cooperation, even though the human and the robot share the same workspace, they work on different tasks. This means that, while in the long run they both work for the same final goal, their tasks have distinct objectives.

Other relevant theme in the context of HMI is how the collaboration is kept secure and reliable. Robots are rigorous machines with great speed, force and power, and these features can escalate a collaborative session into a dangerous situation very rapidly [22]. But robots are not the only risky element in a collaborative session, humans can also mishandle the system which could possibly create malfunctions and unpredictable scenarios. Besides these two factors there are also additional hazards originated from the industrial process such as strenuous demands for the operator related to the task's complexity, duration and repetitiveness, and other environmental threats.

Early on, safety protocols consisted of mostly the existence of physical barriers between the machinery and the operators which resulted in an impaired and very conditional interaction. With the evolution of the industry, new safety protocols were enforced, for example, through the deployment of sensors near the robots in the shared workspace. Nowadays, the safety standard ISO EN 10218 of robots defines HMI collaboration in an industrial environment into four main classes of safety monitoring [46]:

- Safety Monitored Stop
- Speed and Separation
- Power and Force Limiting
- Hand Guiding

Safety Monitored Stop type is adopted when the interaction between the operator and the robot is kept to a minimum. By implementing sensors around the robot, when the human trespasses the safety line defined by the regulation the robot halts its process until it is safe to carry on its function again. Speed and Separation is a more advanced approach when compared to the previous technique. In this case, the robots have a more refined vision system that when detects humans getting closer it instead slows down its movement and if the operator get too proximate, it stops the task's execution altogether. The Power and Force Limiting approach for cobots detects when physical touch is made with the operator and imposes force limitations to ensure that any

kind of collision made could never result in an injury. This is achieved through the use of smart collision sensors and allows for intentional and safe contact between both elements. Finally, in the Hand Guiding mode the robot's movements are controlled by the human via a hand-guided device controller. Even though the robot can still operate in an automatic state, when entering a manual state it only accepts the hand-operated inputs from the operator. As the industry matures, it is safe to assume that new types of collaborative robots are likely to emerge [51].

## 2.3 Multiuser Systems

Nowadays, every person can be entirely connected to other people through digital experiences without any kind of physical limitations or geographical boundaries. With this aspect in mind, the possibility of integrating a multiplayer functionality into the project may significantly improve the collaborative process. Even when both users are unaware of how to solve a task presented to them, according to Dillenbourg [52], if there is a collaborative aspect to the experience, they may not only overcome these challenges imposed on them but also achieve efficient results (Collaborative Learning). Dillenbourg distinguishes the difference between cooperation and collaboration, whereas the first is where "partners split the work, solve sub-tasks individually and then assemble the partial results into the final output", and the latter as when "partners do the work together". In a Multiplayer setting, both types of interaction can be realized simply by setting their initial goal, which makes the whole interactive ordeal remarkably more dynamic and productive than just a Single-Player experience.

Even though the Multiplayer's feature capabilities are compelling, there are plenty of challenges in its implementation [53]. There is a need to reinvent architectures that were previously working accordingly, the use of communication protocols between clients and servers, and also synchronization mechanisms that should enable a seamless interaction among the participants. Furthermore, there are other issues that must be addressed when designing a Multiplayer experience, like the scalability of the system, and the network complexity, which may, in turn, require extensive optimization and also latency mitigation.

Another aspect that should be considered when designing a Multiplayer system is the Client-Server model, which is a networking architecture that forms the backbone of modern data exchange protocols. The client - the end-user device - makes requests for information or services by initiating communication with the server, and the server is tasked to complete the request. Usually, servers are powerful computers capable of processing and handling multiple clients simultaneously. Over time, it was noted that it was possible to achieve higher efficiency by dividing the workload between multiple servers [54] (Distributed architecture), or alternatively using one or multiple of the clients as hosts for communication (Peer-to-Peer architecture). As a consequence of the rapid evolution of the Internet, web services, databases, and other technologies that required a server-based architecture, there was the introduction of on-demand cloud computing to facilitate the work of the software and hardware design.

### 2.3.1 Multiuser Collaboration in an AR Environment

Interaction between multiple users is an important aspect of the development of a system. In an AR environment, this can happen generally in two ways: Face-To-Face (FTF) and remotely, also known as Computer-Mediated Collaboration (CMC) [55]. The first is the most natural and intuitive form of collaboration, since all the users are based in the same physical area it is possible to interact with each other regularly through gestures, speech, and other non-verbal interactions. The implementation of this type of collaboration, albeit more accessible, still needs some preparations beforehand: since the holographic 3D models will be displayed for the entire team, it needs to be positioned in a common place and oriented correctly for everyone, and still be able to work independently if needed. By keeping track of the localization of the users and the transformations of the 3D models, it's possible to position the virtual hologram accurately for each user.

Andres Henrysson et al. [56] developed AR software for FTF interaction through mobile phones and then examined the outcomes related to the efficiency of the collaboration. It was observed that some of the main benefits of FTF collaboration for an AR system are the possibility to see their collaborator, therefore raising the user's awareness due to the presence of visual cues in their communication, and the presence of multisensory output between the pair which facilitated their interaction. Although, during the experience, it was also noted that, even though the participants were facing each other, their main focus was on the phone's small screen, which could provoke an unsafe situation in industrial environments.

On the other hand, remote collaboration, also known as a possible application of a Multiuser Virtual Environment (MUVE), is a server-based environment that represents users with avatars. Depending on the AR gear used, these avatars can be capable of interacting between themselves via gestures and speech as if face-to-face in order to work out together a specific task.

Notably, Wadham Hatem et al. [55], after researching the impact of both types of collaboration in pairs, concluded that CMC was more efficient than FTF. The productivity levels were around 20% better, and the wasted time was reduced by 50%. The reasons for this outcome are purely social: when working remotely, people would usually avoid side conversations, and the transmission of emotional variables was more restrictive which resulted in a similar level of time spent or participation in the task. But even though CMC proves itself to be a more competent way of collaborating in pairs, FTF interaction is still the norm in industrial settings because of the fixed workspace of the machinery, the richness of the data transmitted directly through contact between humans and the high complexity of the tasks' instructions.

Certain tasks can be hard to collaborate on, i.e., if there's a need for one of the users to interact with real elements while the other acts as supervisor. In this case, the user performing the task could, hypothetically, run a Simultaneous Localization and Mapping (SLAM) system in order to forward a model of the physical environment built by the user to the supervisor, while also receiving virtual annotations or real-time advice on the task [57]. Collaborative SLAM is a relatively new field that creates a flexible multiuser remotely shared AR experience by enabling a large group of people co-localize and combine relevant data on the environment. Marco Karrer

et al. [58] and Patrick Shmuck et al. [59] expanded on this theme over the last few years. By equipping the participants, which could be both humans and drones, with visual-inertial sensors, the data from the mapping process collected would be transmitted to a back-end system that would handle it in order to generate a very precise outline of the environment. This research is especially compelling for multiple reasons: this system supported up to twelve remote participants, which is the highest number of users recorded in remote collaborative systems up to date; and due to the very high efficiency and accuracy of the system's design and architecture.

While the concept of AR interfaces was still in early development, Dieter Schmalstieg et al. [60] presented a prototype UI with the capability of allowing multiple users simultaneously. By coordinating a co-located distributed system, it became possible to generate a multiuser collaborative session where each user display was independent of the general context, local, and application. The resultant architecture allowed for the existence of multiple UIs where each user could independently control their own 3D windows. The report also details some of the design philosophies followed for the development of the UI and correctly predicted some of the possible future implementations for systems of the same nature, such as remote collaboration and mobile contexts.

SEAR is a collaborative framework designed and implemented by Wenxiao Zhang, Bo Han, et al. [61] that aims to address the scalability issue of mobile AR systems. The coordinate system of nearby users is synchronized and, since they are close, they are more likely to be interested in common objects for augmentation, so it allows them to share the AR results among themselves. The architecture is incredibly complex since it is trying to accommodate edge-assisted mobile computing, which means that the smartphones used for the experience do not make any of the complex computation themselves - it is instead made on edge servers, which allows for the workload of the mobile devices to be significantly decreased but the reliability on the latency and servers performance is of utmost importance. The system evaluation demonstrates that the SEAR system improves the scalability of AR systems by having most of the back-end on the servers and using the device only for light computation, like exchanging data with the server - mostly data related to the position and orientation of the augmented objects and the users themselves. The system still has several limitations, for example, the application itself is designed for smartphones, and even though it is scalable and should be compatible with new instances of the application, the build behind it does not support easy platform switching for other AR devices. Also, it relies on mobile and opportunistic communication among the users, which may not be available in all environments, such as a manufacturing industry where the noise level is higher and there is more visual clutter.

Ostanin, Mikhail, et al. [62] suggest an architecture that would be very similar to the idea of controlling the robot via their digital twin through AR. The main objective of their project is to program a robot via demonstration, as in, by analyzing and generating cloud points on the trajectory of the finger that is being tracked, it is possible to create a robotic program that mimics the trajectory recorded. Even though their project is not multiuser, they conclude that integrating multiuser capabilities was the most relevant upgrade for their next iteration of the project. INESC TEC has a similar project that aims to program a Universal Robot (UR) via demonstration, made

by Inês Soares, Marcelo Petry, et al. [63]. By taking advantage of this software that is made available to the INESC TEC team, it could be possible to slightly alter it in order to incorporate a wrapper architecture able to withstand a multiuser system. Although, it is also possible that a wrapper architecture may not be compatible, and instead, implementing multiuser could demand a complete rework of the architecture. This is further studied during the system planning and design, in Section 3.

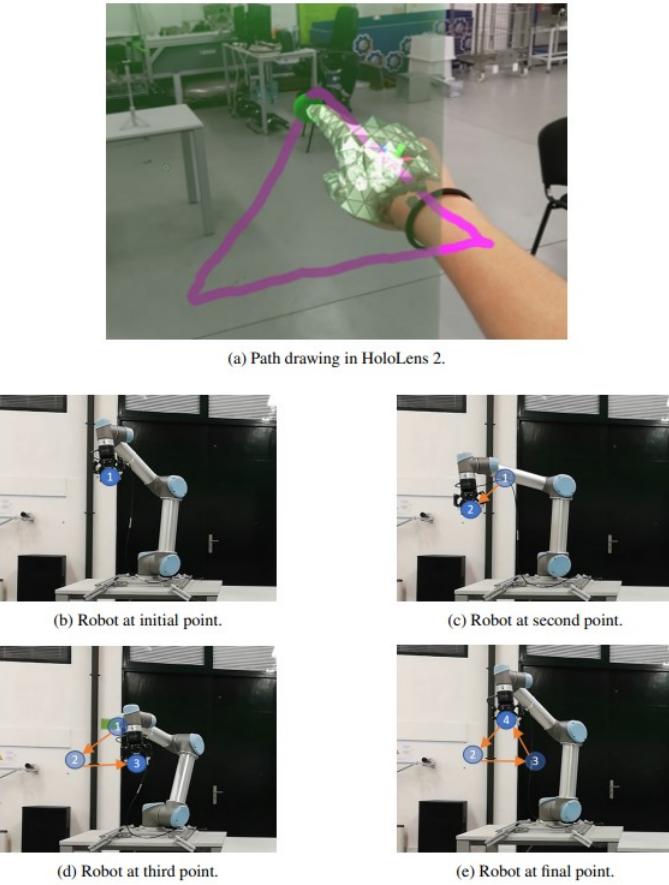


Figure 2.8: INESC TEC’s Programming by Demonstration Software  
(from [63])

The SIGVerse, developed by Tetsunari Inamura and Yoshiaki Mizuchi [64], is a cloud-based system for the VR platform that implemented multiuser capabilities in order to research human-human and human-robot interactions. By using a VR device, the user can participate in HRI experiences via a log-in to their correspondent profile avatar. The system is fully connected to ROS, where each user can visualize in real-time the robot’s movement via a digital twin, and also allows control of the robot via ROS commands computed on the VR application. The SIGVerse allows for remote multiuser sessions since it is fully VR and capable of implementing the server and ROS instance on a cloud server. It aims to provide a simulation environment where real humans can not only interact among themselves but with the robots as well. The system is remarkably flexible and scalable, it also allows the collection of multimodal interaction data quite easily, which could be

beneficial for evaluating HRI techniques and, according to the report, gathering data for machine learning implementation. There are also downsides, for instance, being fully dependent on VR makes the system less intuitive, especially when AR offers the potential for interaction with the real world, and the system also requires manual modification for the robot software due to the original robot-control API, which can be time-consuming. Yoshiaki Mizuchi and Tetsunari Inamura implement an approach that attempts to upgrade SIGVerse by improving the reusability of robot software and eliminating the need for developing device drivers for each VR interface [65]. The application incorporates both ROS and Unity as middleware for VR, and their final architecture, in a general sense, could be used to draw insight into the dissertation project. However, it is still relevant to note that the proposed system is still in the evaluation phase, and further testing and improvement are required to fully assess its potential limitations and performance.

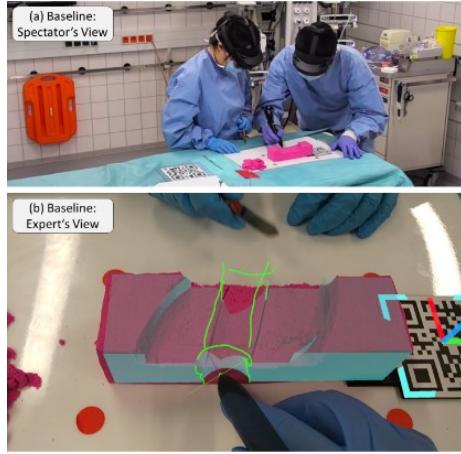


Figure 2.9: Generating a Virtual Model by Tracking the Interaction with Kinetic Sand in a Duplicated Reality Environment (from [66])

Kevin Yu, Ulrich Eck, et al. [66] implement a system capable of simulating real-life multiuser capabilities through the use of Duplicated Reality (DR) and marker-based algorithms. By duplicating the virtual objects and scenes in each user's AR display, the system creates the illusion that users are seeing and interacting with the same virtual content, however, this is not a true sense of multiuser collaboration, since it is actually just a Duplicated Reality. The users are wearing a HoloLens 2, which allows them to see both real and virtual objects, and the system generates a 3D model of a specific task next to the marker - in this case, the 3D model is related to the Pedicle Subtraction Osteotomy medical procedure. A specific user is tasked with shaping a block of kinetic sand to match the desired form, and the system tracks the user's movements and computes what the digital model would look like, which is then displayed to all other users as a virtual augmented element. Even though this characterizes as a multiuser system, it is in a different spectrum of what the project is aiming for. The project focuses on a single user conducting the physical task, while others are capable of visualizing and monitoring their performance via a simulated model that is dynamically generated. Additionally, the report accounts for challenges that the system may face in synchronizing the DR with accuracy when the presence of multiple users.

# **Chapter 3**

## **System Design**

### **3.1 System Requirements**

Augmented Reality technology and its integration into Industry 4.0 has very recently become a trendy topic, with the recent surface of a multitude of systems currently being researched and developed that can, at some part, relate to the project of this dissertation. Using some examples found during the theme investigation in Chapter 2 that can be applied to the project: Tetsunari Inamura and Yoshiaki Mizuchi [64][65] developed a fully VR application where users are able to login to their VR avatar and interact with other users in the same scene, additionally they may visualize virtual robots updating in real-time. SIGVerse's architecture ends up using Photon as the Network package and a SQL database, however, their application is not only limited to VR, but they are also not capable of programming real robots and their computing is completely done in the cloud. Nonetheless, their research is relevant to the implementation of the project's architecture. Additionally, Mikhail Ostanin et al. [62], instead of taking advantage of cloud computing, their architecture uses similar instances of the application for every end-user, but the robot programming is controlled individually on the Robot Operating System (ROS). This may prove to be a future problem, even though it's a simpler architecture, there is no user control which can end up with command overrides, unexpected malfunctions, and other issues that could be fixed through the use of a computing medium as a network manager.

Having these specific systems in mind, and plenty others like the ones researched in Chapter 2, some prerequisites for the system were also considered, such as:

- AR Device - Due to the HoloLens 2 gesture capture capability and its very intuitive nature for the human, the user should be able to grab the virtual model of the robot and move it at will and consequently, the robot should react as commanded. This should happen thanks to the low-level connection between the digital twin and the robot through the use of the ROS framework.
- Face-to-Face Multiuser - A FTF collaborative setting where co-workers sharing the same workspace function together for a specific goal has been demonstrated to bring benefits to

the industrial process. The AR environment should be grounded in the real world which promotes social interaction between the participants, so fellow workers will still be able to communicate in natural ways. Resultant of this implementation is the tantalizing possibility of untrained users benefiting from the perks provided by the multiuser AR experience.

- Spatial Anchoring - Since the application relies on just a 3D model of a cobot, there is no need for image detection complex algorithms, such as the Azure Spatial Anchors. Instead, a basic 2D marker can be placed on the space planned for the FTF collaboration, which should spawn the Digital Twin.
- Database - A requirement that was set since the beginning of the project was the relevant need for a database to store any and all kinds of important information. This would help scalability-wise when integrating new functionalities related to the manufacturing process or the statistics of the machinery. This could prove useful in terms of user control and programming management.
- Robot Interaction - The most relevant of the prerequisites, the need to interact with the robot through its Robot Operating System (ROS). The project's relation with ROS is fixed and one of the main objectives, so the architecture should be compatible with this component in every possible way.

### 3.1.1 Multiuser Implementation Challenges

Even though multiplayer applications can be challenging and demands not only a deep understanding of networking concepts, but also careful planning, testing, and ongoing maintenance, the fact that the system is supposed to work on an Augmented Reality setup presents even more challenges to the common complexities seen on the Chapter 2. In order to start designing the project's proposed architecture, there needs to be an extensive comprehension of the challenges imposed by the enforcement of multiplayer features, especially when in the presence of an already-existent and well-defined system:

- Network Implementation Complexity: Multiple devices or instances of the application need to communicate extensively between themselves in order to achieve real multiplayer. It should be capable of both synchronous and asynchronous functions while transmitting and receiving data. Additionally, there is a need for a network manager to handle all the connections and a way to mitigate latency and packet loss. The server-client architecture also needs to be efficiently adapted using concepts like an authoritative server model or data serialization. These concepts end up adding complexity to the development and testing process.
- Synchronization of the Game State: In a multiplayer environment, there needs to be a consistent game state across all players, as in, the virtual environment, the player's position, and the interactable objects. Additionally, there are plenty of other elements that should be smooth for every user, such as the animations and the game physics. The existence of

latency and network packet delay can ruin the experience by themselves, so the use of an efficient network transport that alleviates synchronization issues is relevant.

- Network Architecture: The network architecture dictates how the packet traffic will be delivered and controlled. Nowadays, the most common approaches are the client-server model and peer-to-peer (P2P), however, each ends up bringing up its own problems and necessary solutions. For example, in a P2P model, the clients communicate directly which can turn synchronous events for a multitude of people challenging, and also there can be issues with NAT transversal technique. Furthermore, the client-server model requires managing a central server and introduces scalability problems.
- Security: The moment an application requires the need for a central server or the external use of a database, the application by itself stops being completely safe in an individual environment, with the possibility of imminent hacks that can range from just server disruption to even source code destruction or data leaks.
- Scalability and Performance: Eventually, if the application is successful, there may come a time when there is a demand to expand the network architecture, either to stand more players or servers, or just manage resources differently. The application must have an initial network architecture's size in mind, however, there should also be the possibility of future scaling to support more assets and still be as responsive, which can prove to be challenging by itself. There needs to be network traffic and bandwidth usage optimization and an effective server capable of managing all the resources.
- Debugging and testing: Debugging multiplayer applications is way more complex than in single-player settings. There can be a multitude of network-related issues, hard-to-replicate or diagnose bugs, and testing the application with multiple instances can sometimes not be enough, since the hardware by itself can create small malfunctions.

AR multiplayer applications rely on spatial synchronization to overlay the virtual objects on the real world which requires precise sensors, low network latency and a very accurate tracking system; otherwise, the virtual elements may be misaligned or even inconsistent. Additionally, since AR systems rely on transmitting a significant amount of visual information, like rendering a 3D model, it can very easily put a strain on the network bandwidth. Finally, it needs to be evaluated how the users will interact among themselves, if there is a need for voice chat, gesture sharing, or text communication, it ends up weighing down on the network bandwidth. In order to achieve a seamless collaborative session between users, there needs to be plenty of research and experimentation.

### 3.1.2 Use-Cases and UML Analysis

Finally, some Use-Cases and UML diagrams were prepared in order to see how each component should interact among themselves. These diagrams are helpful in modeling the system so that every person can share an understanding of how everything works, and it's also beneficial to gather additional requirements that may seem unnecessary or even unnoticed at first. Multiple Use-Cases were made: client logins, server creation, joining online sessions, Digital Twin manipulation, etc., however, the document only includes some of the relevant ones to understand the project's scope.

Initially, no user control was required, since anyone with the application's instance running could actually control the robot. However, by having multiple users collaborating in a single robot, there should be some kind of protocol to avoid overlapping instructions. Even though being face-to-face might reduce this issue by communicating when one is controlling the robot, long-duration collaborative sessions make the operators more prone to eventual mistakes [47]; besides that, having to actively rely on stating when one is controlling the robot will depreciate the user experience along with the decrease of the overall performance since only a single operator is effectively working.

Implementing user controls should not only be beneficial for the potential user control - since any unauthorized user could simply pick up the HMD device and control the robot, potentially causing damage to valuable machinery and products or physical injury to coworkers -, but also for future scalability of the project. Each profile should have a specific authority so that the superior or the operator with the most experience may override the coworkers' inputs. For instance, the head worker may predict an incoming accident and might need to press an emergency stop button. Additionally, user control may create more flexibility and accessibility by letting users customize their own UI, for example, or only get access to the features that they should work with. The UML associated with the login process can be examined in Fig. 3.1.

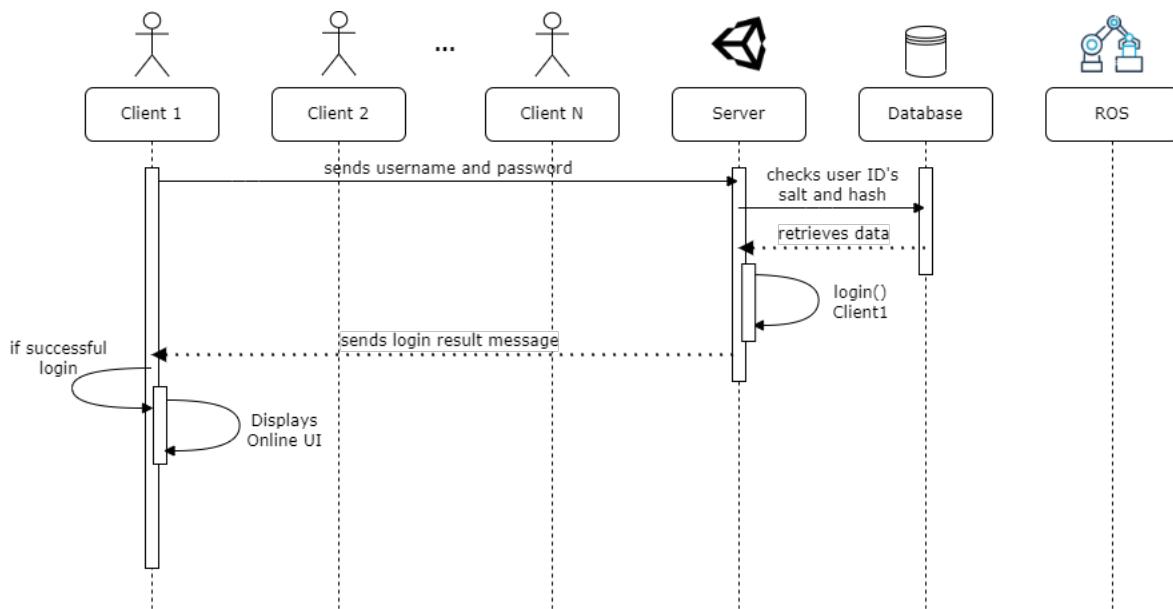


Figure 3.1: Login Sequential Use-Case Diagram

Despite the original aim of the project was to create a single server that could host both a specific robot's Digital Twin and multiple users, if in the future a manufacturing industry wanted to implement multiple servers for their many machineries it could prove to be a nuisance. They would require multiple computers running plenty of servers for each machine in specific which would not only be inefficient but require a main server to control all the sub-servers or just execute and kill the servers by hand. Additionally, there could be the need to run multiple servers with different robots each or just sessions where people shouldn't be collaborating.

Since scalability is always a feature that should be taken into consideration, by taking advantage of the presence of a database and through the use of a single computer, the project instead should be capable of generating a hefty number of Unity servers on the same IP Address just by changing up the port number. Each of these Unity instances, in this project, is called a *session*, since they by themselves belong to one server only but use different ports to receive and distribute data. This technique is known as port multiplexing and is used in networking protocols and applications where multiple connections need to be maintained simultaneously. By utilizing different ports, it becomes possible to handle multiple streams of data or communication channels independently while sharing the same IP address, which enables efficient resource utilization and optimizes the number of servers needed. Although this is a commonly known approach in networking environments, there is not enough data or research online to show if this is possible with the currently available network frameworks, especially since Unity's network libraries seem to not take advantage of this concept. Nonetheless, a capacity for a multitude of servers in a single IP address is a compelling feature. This would also result in the need for an additional master server or just a simple HTTP server that executes the servers as needed/required by the users. The sequential UML shown in Fig. 3.2, explains in a simple way how a session would be created, and how the user could pick a specific session to join.

The final UML, Fig. 3.3, explains how the interaction with the Digital Twin is made locally by every user. The original server creates the Digital Twin as a networked object and shares its current position, angle, and orientation with every user, which in turn can interact with it. The direct relation between the robot and the end-user is cut-off and instead any command given to the real robot by the client goes through the server, where it can be processed, or even denied, and then the real robot performs as required. In comparison to the direct connection, this may create some latency problems where the robot does not act as fast as it could, although, in the long term, the server control could prove useful to avoid critical situations and even support more concurrent users in the session.

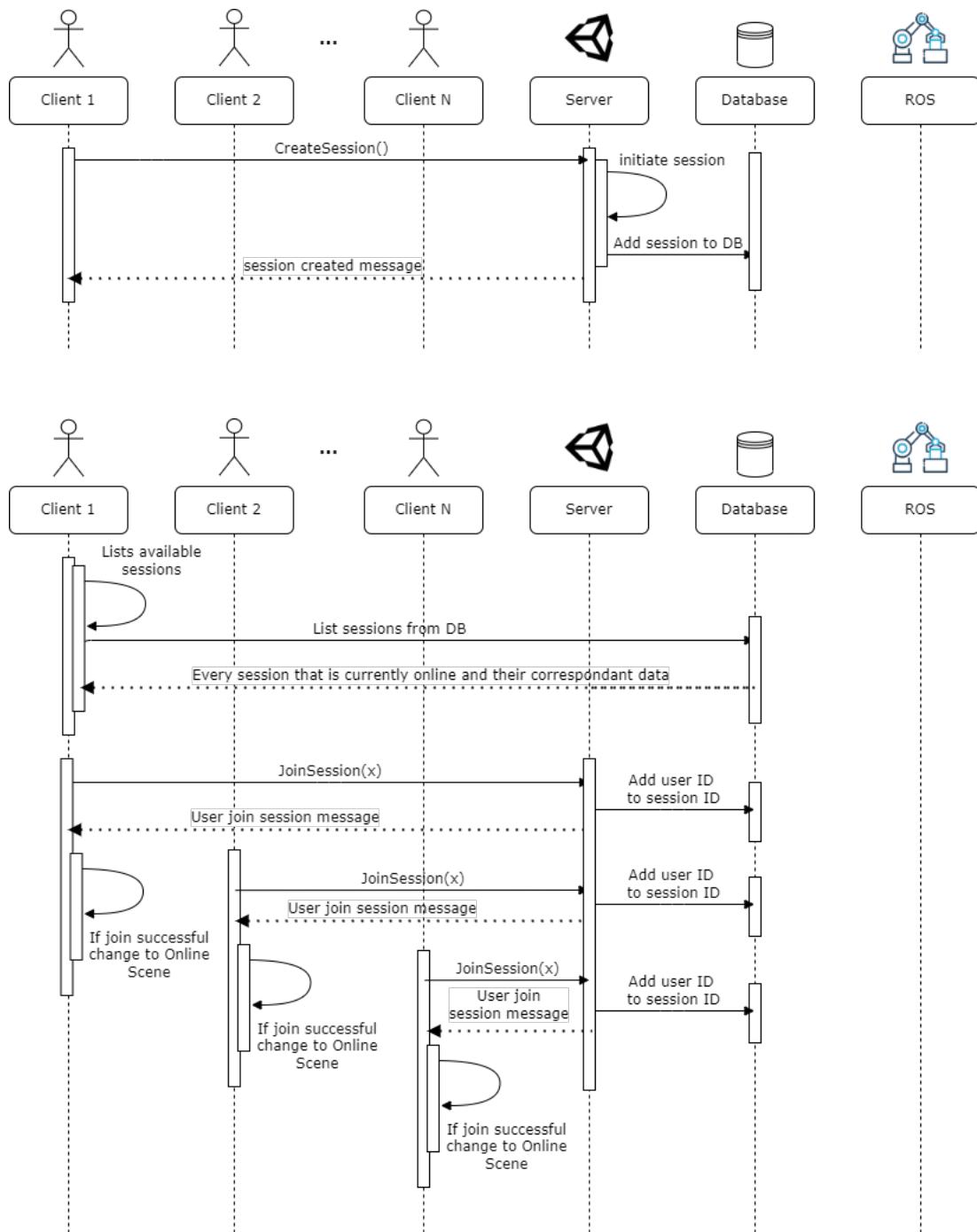


Figure 3.2: Create and Join Session Sequential Use-Case Diagram

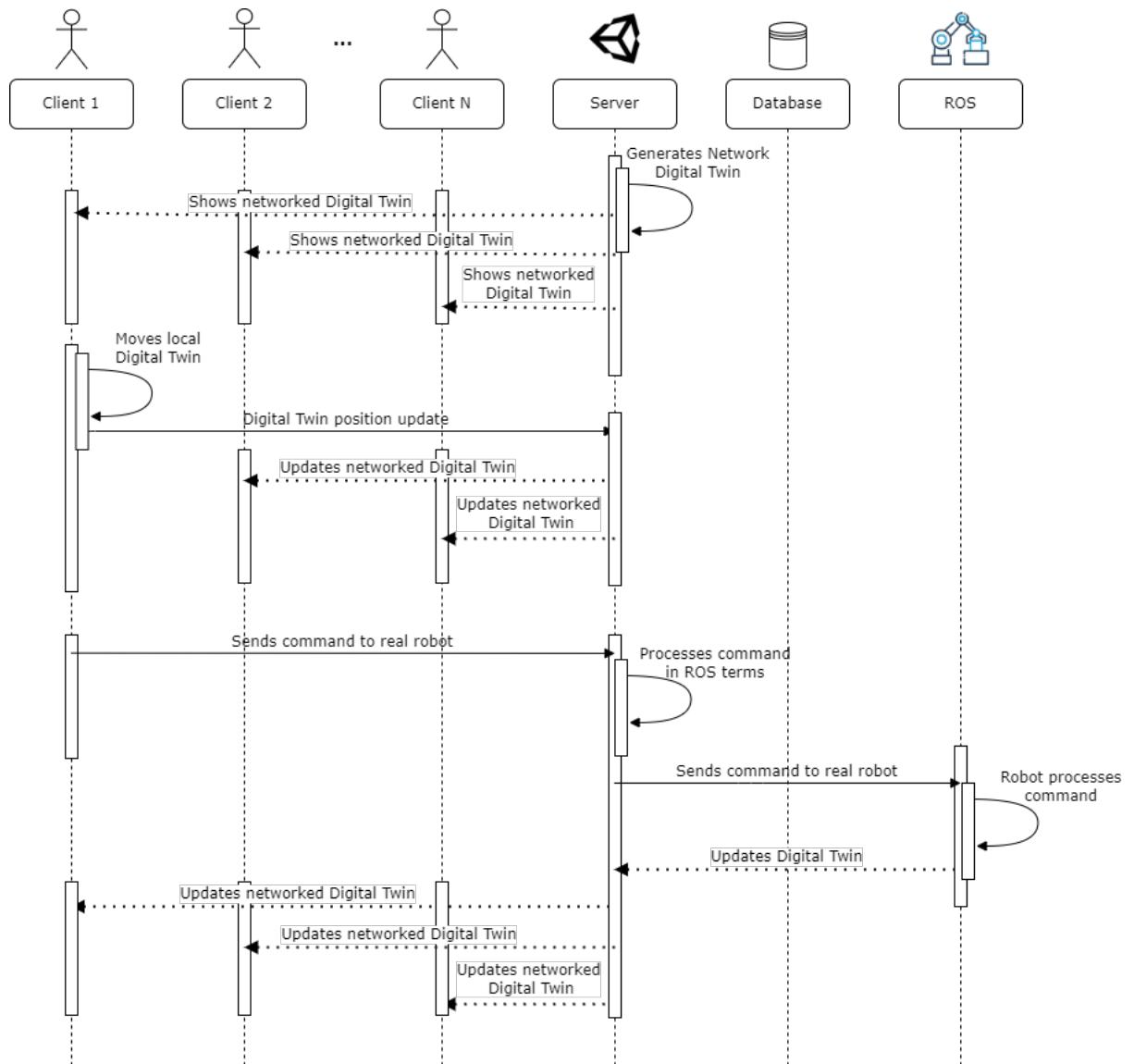


Figure 3.3: Robot Control Sequential Use-Case Diagram

## 3.2 Technologies

In order to complete the project, there will be a need for multiple technological components. This subsection briefly describes some of the most relevant hardware or software required for the dissertation.

### 3.2.1 Microsoft HoloLens 2

The theme of the dissertation arises some needs related to the flexibility and scalability of the project in terms of considering the correct device type for it, for instance, the operator might want to control different robots placed in proximity or multiple distinct users might want to control a specific robot simultaneously. Combined with the easier and more straightforward possibility of future system implementations in additional robots and an eventual cheaper alternative in the long haul, the AR device used will be an HMD smart glasses – the Microsoft HoloLens 2 -, which can be seen in Fig. 3.4.

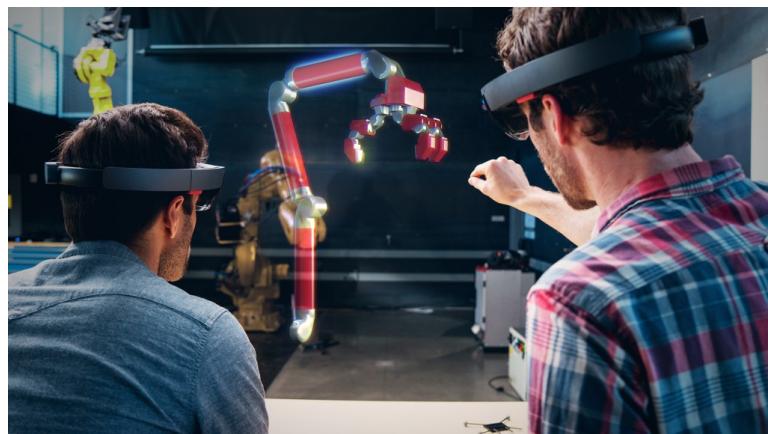


Figure 3.4: Example of an AR environment and the interaction between human and machine via gesture-based inputs on a holographic 3D model of the robot [67]

The Microsoft HoloLens 2 is a wearable AR head-mounted display unit developed by Microsoft and released in 2019 that integrates reality with holographic, computer-generated elements [68]. The first iteration of these smart glasses went on sale in 2016 and made a profound impact on the development of AR applications due to its near real-time high-resolution imaging support, ample computing power and its inherently intuitive, easy-to-use facet. The second generation of the device improves its display resolution, integrates eye-tracking and new gestures support, and a new generation of a Holographic Processing Unit (HPU).

It is categorized as an optical see-through device since the user visualizes reality directly rather than with the use of a camera. The HoloLens 2 is a waveguide and laser-based stereoscopic and its interface is based in perceptive and sensory commands from the user, such as gaze, gesture and voice inputs. The device supports both 2D and 3D applications and was released along with a public repository of auxiliary tools and sample applications in order to encourage development

and research. For this dissertation, the HoloLens 2 is used to display the interface to the operator and interpret the user's input on the holographic, 3D virtual model of the robot as seen in 3.4.

### 3.2.2 Robot Operating System

Robot Operating System (ROS) is a open-source aggregation of software libraries and tools for robots that first released in 2007. It is capable of providing services expected from an operating system, such as hardware abstraction, inter-process communication (IPC), and low-level device control [69]. By providing several libraries, services, and tools for robotic applications, it also enables for an easy implementation of robot functions, such as locomotion, vision, navigation, sensing, and many others. Additionally, it also acts as a global platform for development and public distribution of frameworks related to the robotic field between people.

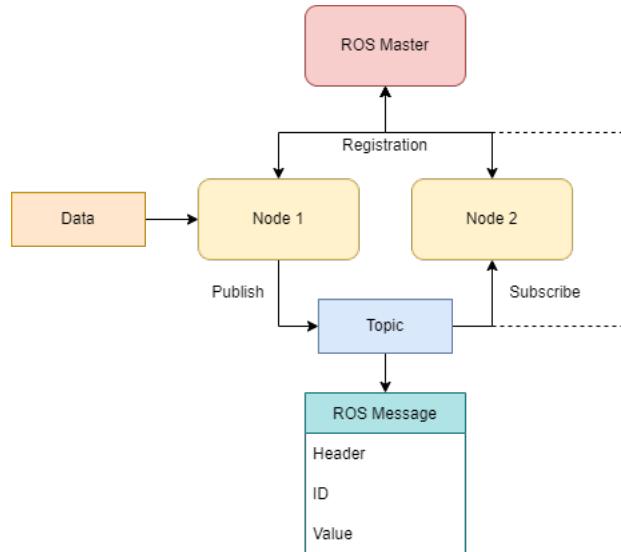


Figure 3.5: ROS architecture for a subscription-based system

The communication in ROS was designed to be nonrestrictive and flexible. The figure 3.5 provides a straightforward approach to the ROS architecture design. The ROS Master provides the naming and registration services to all the other nodes in the system. Its main purpose is to allow ROS nodes to locate each other and communicate peer-to-peer by keeping track of all the publishers/subscribers to topics and services. A computational process is known as a Node. Each node is bound to a single process and, if the node is subscribed by another, it is able to communicate through unidirectional messages. The communication in ROS is very diversified which facilitates data format conversion; there are three special kinds of communication:

- **ROS Topics** – This method is mainly used to send continuous data streams between nodes. Since a node is not aware of the other nodes in the system, in the case it is interested in data monitored by another process, it needs to subscribe to a topic. Contrarily, if the node wants to share data with others it must publish the data by creating a topic. A topic can have several subscribers and publishers.

- ROS Services – By creating a very simplified synchronous client-server communication between nodes, it allows for a better way of sharing information in distributed systems since having too many publishers/subscribers in a single topic can become inefficient. This is achieved by having a node acting as the server for the service. When a node sends a request to server, it then redirects the request to the target node while waiting for the response. When getting the response, the server delivers the message to the initial node. Services are especially useful for tasks done only occasionally and that take some time to complete.
- ROS Actions – Even though it is based on topics, it is a more complex format of message with an implementation of client-server communication. An action server receives a goal request from a certain node, and then initiate the processing on the corresponding nodes required to accomplish that goal, and when the objective asserted is completed it then sends the result to the initial node. Due to this delay between the request and response, actions are considered an asynchronous type of messaging. Also, an action can be cancelled any time during its execution.

ROS as a platform provides numerous packages composed of programming files, *executables* and documentation. Even though ROS was not designed initially as a language-neutral operating system, it is possible to easily implement additional modern programming languages through the use of additional frameworks to convert the files into a preferred programming language. For this dissertation, due to project's heavy nature in AR it was required to resort to the ROS Sharp software libraries in order to enable communication with ROS through C# [70].

### 3.2.3 Unity

Unity, also known as Unity 3D, is a game engine developed by Unity Technologies. It is a C# programming environment that enables the development of 2D, 3D, AR and VR applications with high-fidelity graphics. One of the main characteristics that differs Unity from most other game engines is the facilitated transition between platforms and compatibility between multiple devices. This engine is intended to be used as the design and implementation tool for the user interface for the AR device.

To be used conjointly with the Unity software is the *MRTK-Unity*. *MRTK*, which stands for Mixed Reality Toolkit, is a project published by Microsoft that assists in the development of MR apps by providing a set of components and features. According to the official Microsoft documentation page, the *MRTK* can [71]:

- Supply the developers with high diversity of UI building blocks for spatial interactions
- Accelerates the prototype cycle by providing an in-editor simulator for the implementations
- Operate as an extensible framework, allowing developers to swap out core components
- Support multiple platforms

The *MRTK* package is mainly used for UI creation through its smart input controls, by simplifying commands like gestures, gaze and other controllers. It is also capable of spatially mapping the environment around the user and provides some features to the HoloLens 2, like the spatial audio and eye tracking. Over the course of the project, a new and completely revamped version was released: the *MRTK3*, which, due to timing constraints, wasn't used for the system.

And, the other main packages that will be used can be obtained in the Unity Robotics Hub, which is an open-source repository with a multitude of tutorials, tools and resources that can be used to simulate and work with robots. The packages acquired from the Hub are mainly the *ROS-Unity Integration* and the *URDF Importer*, but new packages and tools are being developed at the time, for example, the package *Visualizations*, that lets the user visualize incoming or outgoing ROS messages. The *ROS-Unity Integration* is a software package that combines two popular and powerful tools - Unity and ROS - by providing a bridge between them, which in turn allows developers to incorporate immersive robotics functionalities in their Unity scenes. These applications can be used to training, testing, prototyping and visualization of real-world sensor data. On the other hand, the *URDF Importer* is a tool that allows the import of URDF (Unified Robot Description Format) files in Unity. URDF is a file format, XML-based, used to describe the structure and kinematics of a robotic system.

### 3.2.4 Mirror Networking

As multiplayer experiences keep growing, not only in the capacity of concurrent users experiencing them but also in their infrastructure's robustness and efficiency, the number of network libraries (also referred to as netcode) also keeps expanding to keep up with this surge. Unity offers various network libraries capable of providing real-time communication, synchronized game states, and smooth connectivity. Since this may be one of the most relevant components to be chosen for the project's architecture - it needed to be compatible with other components, have a decent set of features for future implementations, ease of integration, and, overall, a good performance -, there needed to be careful research and planning before picking the library.

High-Level API (HLAPI) network libraries in Unity are used to facilitate network communication among users, in a way capable of ensuring seamless data transmission and low latency, by providing the developers with tools, communication protocols, and other APIs to create the multiplayer application. At the moment, the most popular network frameworks are *MLAPI*, *DarkRift2*, *PUN*, *Mirror* and *UNET*. The Unity Blog [72] makes a compelling comparison between some of these libraries. The ones that took into consideration were the following:

- Photon Unity Networking (PUN)<sup>1</sup> - one of the most widely known netcode library due to its simplicity and easy integration. Has cross-platform integration and hefty features. Offers cloud-based multiplayer services for long-distance experiences and has a very large online community and plenty of documentation and guides. An additional big plus of this library is that Microsoft's *MRTK* has an extensive repository with tutorials, guides, scripts, and scenes

---

<sup>1</sup><https://www.photonengine.com/pun>

with PUN multiplayer integration. However, it is not free to use if the scope of the game ranges from medium to large or if you intend to use the cloud feature. Also, it relies on an external interface for key activation and control, which means the project may become bound and dependent on external software.

- Unity Networking (UNET) <sup>2</sup> - it is the native networking solution provided by Unity, so it's already built-in and tightly integrated with the engine. It provides all the basic features for synchronization, RPC (Remote Procedure Call) system, and even network discovery; supports both peer-to-peer and client-server models. The disadvantage of this library is that it was deprecated and no longer maintained by Unity since 2018, which means that it could have compatibility issues with newer Unity versions, therefore it is not ideal for long-term projects but it is perfect for prototyping and fast testing.
- Mirror <sup>3</sup> - it offers a very optimized and flexible network based on the deprecated UNET. It is very lightweight, with a focus on performance by having a very efficient bandwidth usage and optimized netcode. Since it is based on UNET, it has all its previous features and even more, like a better RPC system, animation and events synchronization, and authoritative client-side prediction. However, its community is still growing and, in comparison with other netcodes, it has a steeper learning curve with more manual configuration and even mid-level setup, whereas other libraries could keep their solutions at the high-level programming. It is, nonetheless, a very suitable option since it is also completely open-sourced and there is continuous development and improvement.
- Netcode for GameObjects (NGO) <sup>4</sup> - an open-source network library that focuses on Game Object synchronization, as the name suggests. Like Mirror, it is remarkably lightweight by providing a very simple API for synchronizing objects across the network, which means that it is ideal for small projects or prototyping in general. The downside is that, since it is so simple and rudimentary, it means that it is lacking more advanced features for complex multiplayer games and may need some manual tweaking for it to be optimized. Also, there is very limited documentation and community support, which can pose a challenge for developers in need of assistance.

There are other notable network libraries besides the ones mentioned previously, each with its own special features and unique target application style. The Table 3.1 is adapted from the Unity Blog [72], although, slightly changed since the blog post has aged and some of the network libraries expanded or got even reworked, for example, *MLAPI* was remade into *Netcode for GameObjects (NGO)*.

---

<sup>2</sup><https://docs.unity3d.com/Manual/UNet.html>

<sup>3</sup><https://mirror-networking.com/>

<sup>4</sup><https://docs-multiplayer.unity3d.com/netcode/current/about/>

Table 3.1: Summarized comparison between the different netcodes (adapted from [72])

Netcode	Stability	Ease-of-use	Performance	Scalability	Cost
NGO	+	±	+	±	Free
DarkRift 2	+	-	+	±	\$100
Photon PUN2+	±	+	±	-	Free for up to 20 players
Mirror	+	±	+	±	Free
UNET	±	+	±	-	Free

During the project's execution multiple netcodes were experimented with: started with a UNET prototype that moved on to the NGO package, however, the lack of long-term support by the NGO developers and the fact that updating to a more recent NGO version could completely break the application made its use a future liability, which ended up being scraped from the project. Mirror was the final choice since it has all the features NGO has and is completely capable of providing a final result with great performance, low latency, multiple transport protocols compatibility, and plenty of other compelling factors. In addition, even though the system should support multiple different types of transport, one of the main featured transport protocols on Mirror is the KCP transport<sup>5</sup>, which offers reliability and high-performance communication which should enhance the networking capabilities of the system. It is designed to minimize latency by optimizing the transmission of packets and estimating in real-time the network congestion, which it is also able to temporarily solve thanks to its very complex algorithm. It has other noteworthy additions like packet re-transmission and forward error correction mechanism, and the ability to have its parameters fine-tuned in order to optimize the transport for different network conditions. Other network transport solutions will be analyzed during the actual implementation of the system.

Towards the end of the project, a new network library was openly published that is considered the most state-of-art netcode for Unity at the moment: the Fish-Net<sup>6</sup>. It is not only even more optimized when compared to the present competition, but it has plenty of more features that others do not have, like lag compensation, powerful additive or stacked scene management, server sharding and load balancing, and many others. Nonetheless, since the project was almost concluded at the time, the Mirror framework continued to be the one chosen as the network solution.

### 3.3 Proposed Architecture

Throughout the project's life, the system took many different forms due to a lack of foresight, technology incompatibilities, and requisites/scopes changing, which resulted in multiple components and technologies being swapped or reworked. In spite of that, in order to accomplish the goal of the dissertation, the final framework architecture that was proposed for the system is shown in Fig. 3.6.

<sup>5</sup><https://github.com/skywind3000/kcp/blob/master/README.en.md>

<sup>6</sup><https://fish-networking.gitbook.io/docs/>

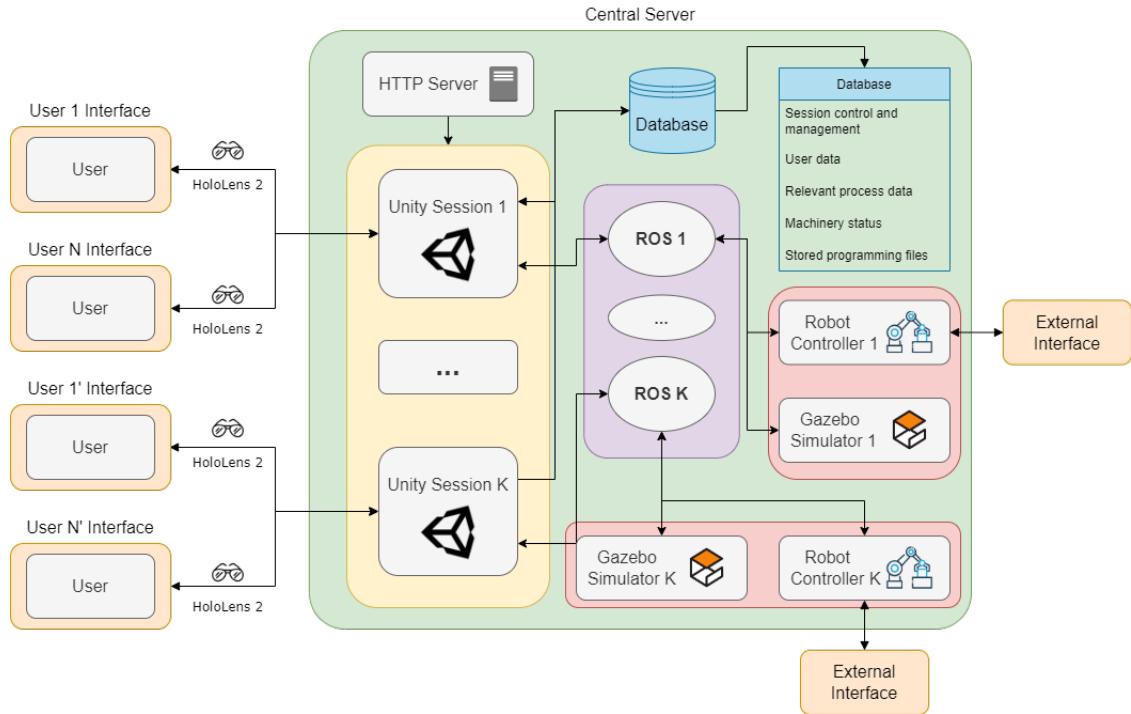


Figure 3.6: Overview of the System’s Architecture

The main components of the system are the users’ interfaces which represent what the user is currently visualizing through the AR display device: mainly the User Interface and the Digital Twin. Each user’s UI is completely standalone and independent, although, other users should be able to see if the worker is using his interface. The holographic model they are interacting with is also completely separate from others, but, if the user makes any change to his personal Digital Twin it should update the server that, consequently, updates every other user’s hologram. The central server is responsible for all the hard computing and data format conversion, it generates Unity Sessions on request, makes database fetches and updates, manages users, and controls or simulates the robot, depending on the type of collaborative session. It’s important to notice that multiple ROS instances or nodes (depending on the ROS implementation) need to be running to control any individual robot, where the Gazebo Simulator is used for the virtual simulation of the robot, and the robot controller to manipulate an actual, real robot. This type of client-server architecture is a common approach for a distributed system, especially considering the multiuser aspect. A simpler-to-understand architecture was designed with only one session in mind, as seen in Fig. 3.7. In this altered architecture there is only one server that connects to one robot alone.

The UI is personally generated on each user’s end and displays both the control interface and the AR model of the robot. The user may interact with the model in order to control the machine if his user profile’s authority level is sufficient. For each user, there should be a unique and independent data handler in order to maintain the aspects of independence and individuality from the collaborative AR interface. In order to achieve this, in the project, only the Digital Twin is collaborative, and the rest is independent of other users’ interactions. The interface and

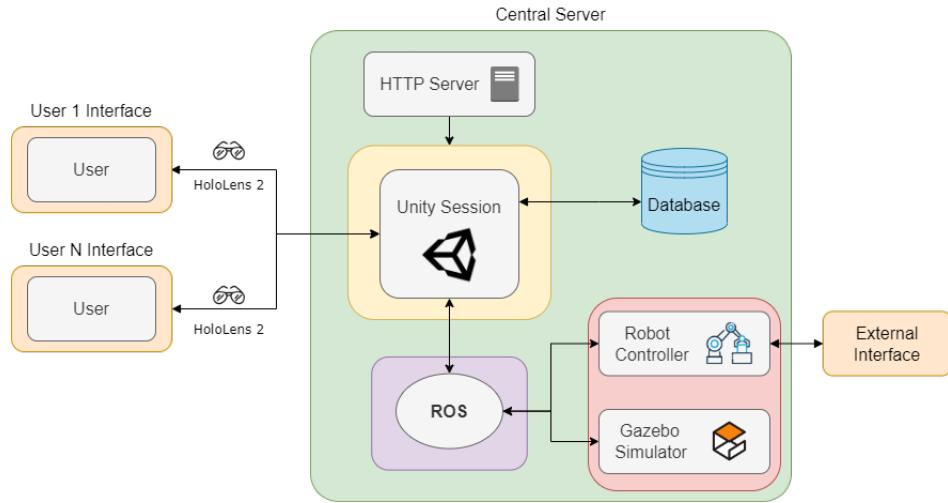


Figure 3.7: System's Architecture with One Instance of a Session

the virtual model that is shown on the HoloLens 2 are developed with Unity3D combined with the Microsoft framework Mixed Reality Toolkit (MRTK) and then compiled on the client-side. Additionally, a network framework must be selected to synchronize the position, rotation and scale of the networked virtual robot model.

The commands imposed by the user on the robot are regulated by the HoloLens2 and processed as a sequence of waypoints in a C# format. This array of instructions is the effective programming file that the robot controller needs, however, it is not initially compiled in a compatible programming language, so the file is handled by the Unity Robotics Hub in order to convert it to a comprehensible script for ROS and forward to the robot's IP address. After ROS receives the script, it is processed and then sent to the machinery hardware either via Ethernet connection or Wireless. There is also the possibility of an external interface directly connected to the robot controller. The robot accepts the input if valid and starts the task's execution; relevant data from the execution, such as velocity, joint orientation and other sensor information, should also be updated in the database so that the user can monitor the task in real-time, however controlling these statistics ends up not being the main scope of the project. Alternatively, if the cobot is not available for any motive, the Gazebo Simulator can be used in its stead through the use of a Virtual Machine. This entire connection should work both ways, as in, if the cobot is being controlled by a coworker, after the machine receives the instructions and starts its operation the user should also be able to view the holographic model of the robot move.



## Chapter 4

# System Implementation

The actual implementation of an Augmented Reality multiplayer system that is used to interact with virtual and physical robots and has an integrated database functionality is, by itself, a relevant milestone when considering how immersive and interactive the experience can be. This section focuses on the intricate and practical process of software development and system engineering with the objective of creating an AR multiplayer experience capable of all the features required and, also, with a favorable performance. It should first be considered that even though, theoretically, the system is already designed and planned to work, as seen in Section 3.3, for a multitude of reasons the actual implementation can easily fail, for example:

- Insufficient or poor requirement analysis - This aspect should be ideally mitigated due to the previous extensive research on not only the technologies - Chapter 2 - but also the prerequisite and objectives of the system - Sections 1.3 and 3.1.
- Technical challenges - This element might be the most coarse and unpredictable one, even when performing a thorough research and testing the technologies individually, in the end, simple packages, assets, scripts, or abstract concepts can turn out to be incompatible for many reasons, like small feature details and interactions, conflicting architectures, versioning differences, dependency conflicts, and even coding conventions. In this situation, it could completely make the success of the project impossible and the only way of mitigating this impact is by having multiple possible framework solutions available and creating a flexible enough architecture that can adapt to new and different additions.
- Requirements changes and scope creep - The dissertation itself suffered from this aspect during the development process: whereas initially, the aim of the project was broader, for example, there was the possibility of extending the available robotic simulation with new programming or AR features, or, on the other hand, expand the database and develop a software that was compatible with the simulation for a complete manufacturing process analysis with KPIs and machinery evaluation; in the end, the focus became mainly the collaborative AR aspect and the multiplayer implementation into the robotic scene.

- Insufficient technical knowledge - This final prospect is to be expected when the person in charge of the project has limited experience with the technologies being used for the system. In order to avoid this situation, the person should first explore, analyze and experiment with the technologies to get a better overall understanding.

The following subsections delve deeper into some technical aspects of each of the main components, where it is explored some of the challenges, design considerations, and implementation strategies. In order to achieve a more orderly analysis, a very simplified application flow UML, Fig. 4.1, is used as a reference, where each screen and its back-end logic will be evaluated individually and in detail.

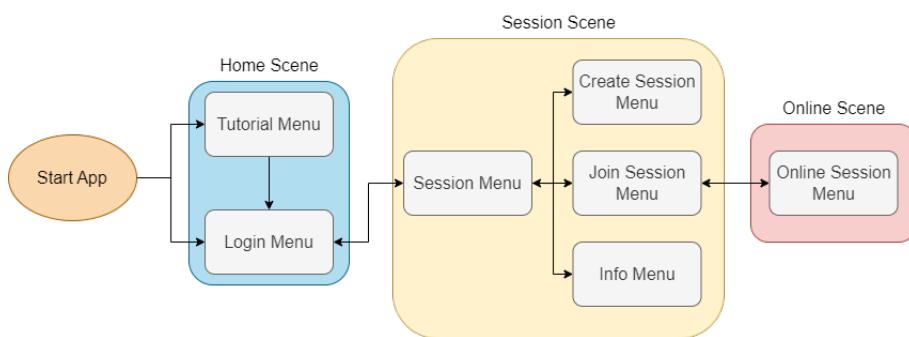


Figure 4.1: Simplified App Flow UML

The main elements analyzed in the following sections are related to the setup of the development environment; the main challenges of the UI design, the choices made to optimize the User Experience (UX), and how the iterative design process evolved the UI to its final look. It is pertinent to note that since multiple components were implemented simultaneously and the synergy between them is too relevant to be evaluated individually, instead it will be analyzed each major scene of the application flow (Fig. 4.1) and how they incorporate the technologies - this way it is possible to achieve a more organized and sequential overview of the application. Finally, as the conclusion of the implementation process, the application deployment on the HoloLens 2, its challenges, and some of the changes to accommodate the features on the physical device are reviewed.

## 4.1 Environment Setup

The foundation for a successful project execution is a well-configured computer environment. This section provides an overview of some of the key components involved in setting up the computer environment in order to work on the many technologies referenced in Section 3.2, like web and database development, augmented reality, application development, and robotics.

Central to the computer environment lies XAMPP<sup>1</sup>, a popular web/PHP development package capable of integrating an Apache server and database management. The Apache environment was

---

<sup>1</sup><https://www.apachefriends.org/>

set up by deploying an Apache server with a Secure Socket Layer (SSL) on port 4443 and with an HTTP port on 8080, coupled with MySQL on port 3306. Since there the computer didn't have a registered Domain Name System (DNS) name, it was used the localhost IP address for the server allocation. Moreover, there is the inclusion of phpMyAdmin<sup>2</sup>, a web-based interface that can be used to streamline and facilitate the management of the MySQL database. Therefore, XAMPP assumes a pivotal role in the project as the medium between the application and the database, since the application needs to run PHP scripts through the Apache and MySQL servers in order to create, update or delete database tables.

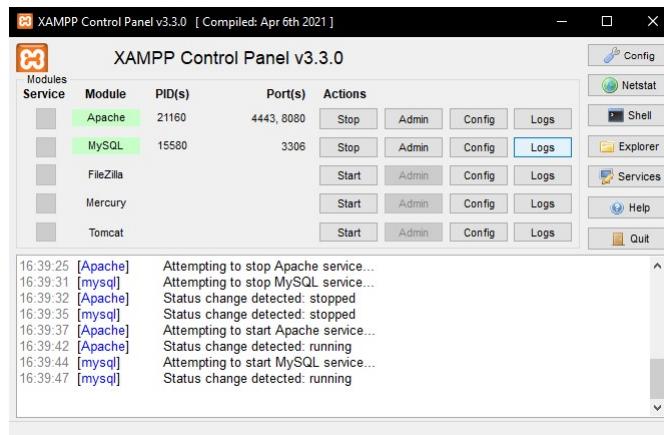


Figure 4.2: XAMPP Control Panel

Additionally, it was also used VirtualBox<sup>3</sup> in order to create the robotic environment, since by configuring and deploying an Ubuntu Virtual Machine (VM), it is possible to cater to the requirements of the ROS-based simulations. A network bridge adapter was created on the VM configuration in order for the application running on the Windows 10 environment to communicate with the Ubuntu-based ROS demonstrations. The Linux VM supports the integration of sophisticated simulation software like Gazebo<sup>4</sup> and RViz<sup>5</sup>, thereby enabling a flexible and optimized ecosystem for robotics through virtualized experimentation. It is relevant to note that the INESC TEC's research team had previously made significant ROS tech demos that could be used as a basis for the project by integrating and modifying it as needed, and the Ubuntu Virtual Machine image was openly available for the team's investigators.

Through the Unity Hub<sup>6</sup>, it was installed a compatible iteration of the Unity game engine - an open-release version that would be compatible with all the technologies, packages, and APIs planned on being used. It should be considered that Unity can have backward-compatibility issues, so the chosen version is one of the most recently released as of the date - 2021.3.10f1. Some specific modules were also installed in conjunction with the Unity engine, such as WebGL, Universal Windows Platform Build Support (UWP), Visual Studio 2019, and Windows Build Support

<sup>2</sup><https://www.phpmyadmin.net/>

<sup>3</sup><https://www.virtualbox.org/>

<sup>4</sup><https://gazebosim.org/home>

<sup>5</sup><https://wiki.ros.org/rviz>

<sup>6</sup><https://unity.com/download>

(IL2CPP). The WebGL module would allow for the deployment of the application on the web, while the UWP catered more to Windows-based devices, like the HoloLens 2. Visual Studio 2019 would just be the IDE used to program the C# scripts for the application. Some of the most relevant assets and packages installed were:

- Microsoft.MixedReality.OpenXR 1.3.1
- Microsoft.MixedReality.Toolkit.Examples 2.7.3
- Microsoft.MixedReality.Toolkit.Extensions 2.7.3
- Microsoft.MixedReality.Toolkit.StandardAssets 2.7.3
- Microsoft.MixedReality.Toolkit.Foundation 2.7.3
- Microsoft.MixedReality.Toolkit.Unity.Tools 2.7.3
- Unity.Robotics.ROS-TCP-Connector 0.7.0
- Unity.Robotics.URDF-Importer 0.5.2
- Unity.XR.OpenXR 1.5.3
- Mirror 78.3.0
- Vuforia Hololens 2 Sample 10.15.4

Finally, as a way of simulating the application and how it would interact with the real HoloLens 2 device, it was installed the HoloLens 2 Emulator. Even though Unity's game scene could already simulate the 3D XR environment seamlessly, the emulator was used mainly as a testing and debugging asset, since there wasn't a need for the physical device.

## 4.2 User Interface Design Choices

Developing an AR interface presents unique challenges due to the novelty and complexity of the technology. As a relatively recent concept, AR interfaces require innovative design approaches and careful consideration of user experience in order to achieve seamless integration of virtual and real-world elements, and since there is a lack of established design principles at the moment, there are plenty of technical or design problems that must be addressed with a careful approach:

- The third dimension - Unlike traditional interfaces, which are confined to the second dimension, AR adds a third dimension, which arises the need to design the UI with depth in mind. Even though this effect can enhance the sense of realism and create more natural interactions with the virtual objects, it needs to be accounted for, which adds another layer of complexity, especially if using spatial mapping.

- Color palette and background - The color choices for the interface need to contrast well with the physical environment to ensure visibility and a decent user experience. This means that the environment in which the user sees the interface matters greatly since there could be variable lighting conditions or dynamic and cluttered backgrounds. A potential fix for these potentially unpredictable environments would be a transparent UI, which allows users to maintain a clear view of the real world, but this could not only turn into visual clutter but also blend too much with the environment, making legibility degraded and creating a confusing level of depth.
- Lighting challenges - Changes in lighting intensity, color temperature and shadows can easily affect the perception of virtual content, which impacts the visual quality of the AR interface. In order to maintain the illusion of realism, there should be a dynamic adjustment that adapts to the lighting condition.

With these challenges in mind, MRTK facilitates and simplifies the development of the AR interface through its very extensive built-in components and tools. For instance, MRTK provides already spatial mapping capabilities that enable accurate occlusion and intuitive interaction with both the physical and virtual environment, which creates immersive and responsive interfaces; it also optimizes lighting and shadow management tools in order to calibrate virtual objects.

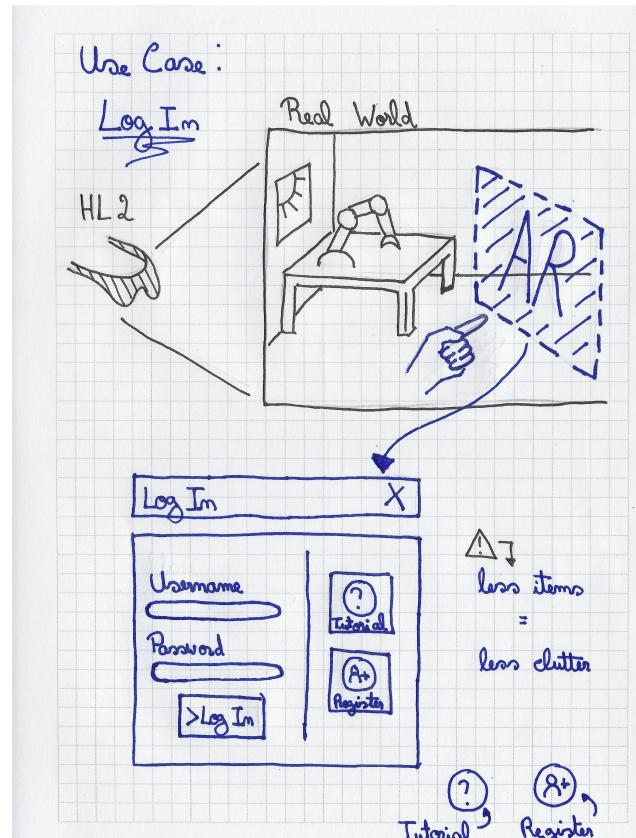
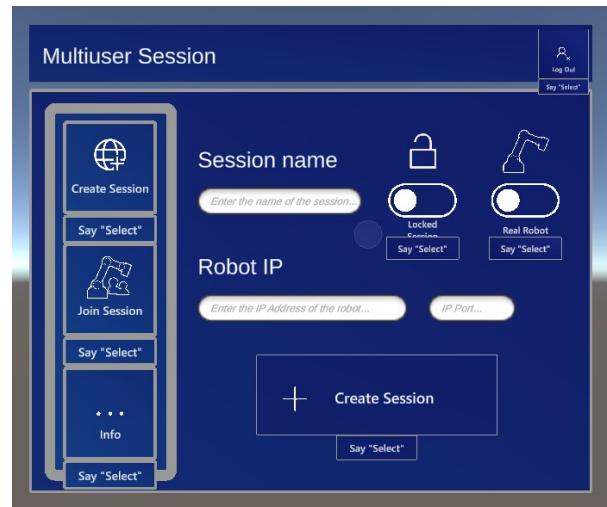


Figure 4.3: Low-Fidelity Prototype of Log In UI

But as previously stated, there is no established design process or protocol for AR interfaces yet, so most of the decisions ended up being either personal choices or trial and error with the aim of achieving a proper user experience. To accomplish this, an adapted iterative UI design process was used and the employment of low-fidelity and high-fidelity prototypes for testing was also adopted. Iterative UI design is an approach that involves creating and repeatedly refining a UI through cycles of design, evaluation, and iteration. It focuses on understanding the user needs and difficulties, testing concepts, and incorporating feedback. By adapting this process, the feedback came from personal opinion and experience, and the number of cycles was lower, allowing for some flexibility between implementations. Finally, the low-fidelity prototype is a paper-based drawing, such as Fig. 4.3, while the high-fidelity prototype is an MRTK demo with all the text, input fields, and buttons, but not actively working. It is possible to see the difference between the two iterations of the design process in Fig. 4.4, where the first iteration had more inputs and had a less organized look, while the last one improved on most of these aspects.



(a) First iteration



(b) Last iteration

Figure 4.4: First and Last UI Iteration Cycle

Even though multiple colors were tested and MRTK has more materials and textures to work with, the color palette ended up being the default theme since it not only gave the most professional outlook but it was also the least distracting from the real world. Finally, the third dimension addition which could create depth perception problems was solved by personal qualitative judgment and experimental iteration. This can be seen in Fig. 4.5, where the third dimension is used to reproduce a depth that increases the user experience by making the interactable elements more prominent due to MRTK creating fake shadow renders on the canvas and, overall, facilitating user navigation.

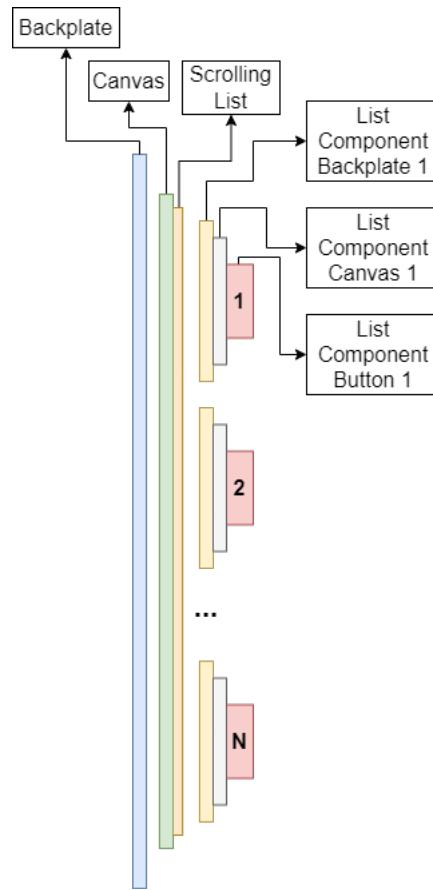


Figure 4.5: Side-View of the AR Interface of a Scrollable List

### 4.3 Home Scene

As seen in Section 3.1, there was a need for both user control and an early establishment of database usage in order to provide a better user experience, and how would it interact with the rest of the application. So, the focus on the Home Scene was the implementation of user registration and login, through both web and inside the AR application. The Fig. 4.6 explains visually the logic behind the authentication and validation process in a simpler way.

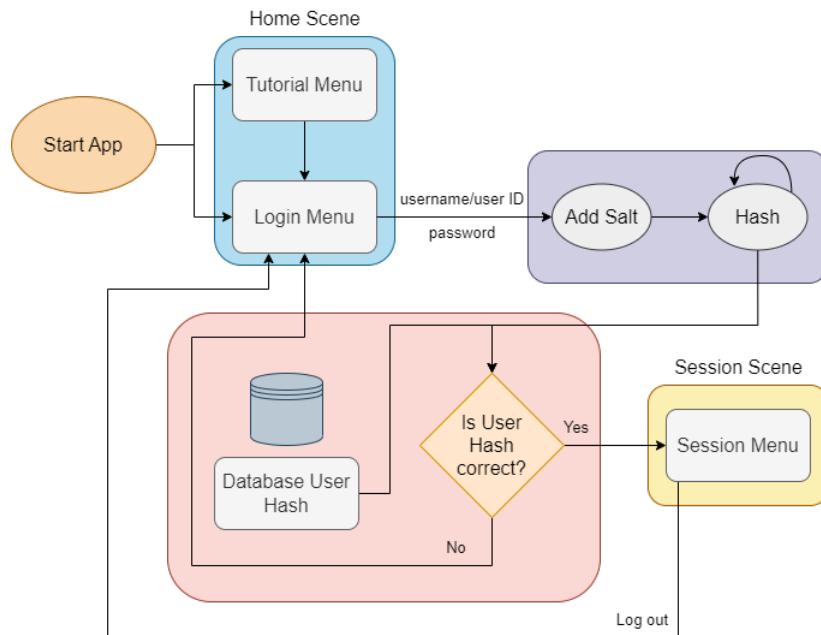


Figure 4.6: Simplified Home Scene Flow UML

Since one of the aims of the project was to attempt a complete release of an end product, there was the need to implement a very streamlined user control protocol but also safe enough to avoid any kind of database leaks. While initially, it was possible to register new users through the AR application, that feature was later removed and replaced by a clean web-only user registration through an already-registered user. That way, when a new operator is allowed to use the HoloLens 2 and manipulate a real robot, their account is dynamically created on the web by their superior, which may mitigate unsupervised physical damage to the hardware and optimize the registration process since it is faster to do it on the computer. This arises the demand for the establishment of some kind of hierarchy between workers, which may prove itself useful for later features, such as command overrides and session ownership. At the moment, the registration website is set to the server's IP (localhost) HTML index on port 8080, although there is the possibility of integrating it on an official company website or creating a DNS name and allocating it online.

The proposed hierarchy is a very straightforward one, as seen in Fig. 4.7, where the first and only account available initially is the administrator, which ideally should be given to the head of the technology department in the manufacturing industry or a Human Resources director, for example. The administrator is the only user with the maximum authority, but can, however, generate new

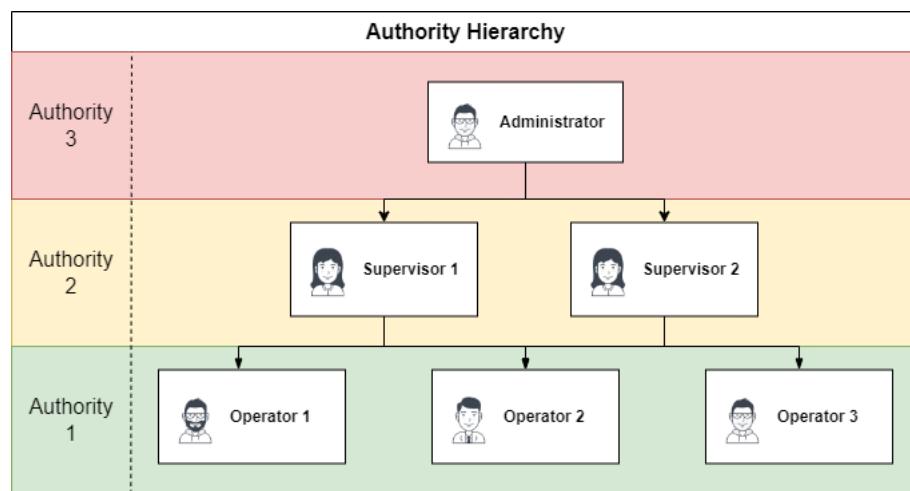


Figure 4.7: Authority Hierarchy

accounts for the industry workers. He can choose the authority that each specific user should enact depending on the worker's experience, for example. It should be noted that any user can log in to the website and, if their authority allows, create new accounts, but the new users always have a lower authority than their creator. For instance, an administrator can create accounts with authority 2 (supervisor) and authority 1 (operator); on the other hand, supervisors can only generate operator accounts with an authority of 1. Operators can still log in, even though there are no features for them on the website at the moment. In the future, the website can be further developed in order to accommodate more features, such as allowing users to create or list all online sessions through the web, display a history of their online work, or add the possibility to check the machinery or manufacturing processes' status.

Username/ID:	<input type="text" value="admin"/>
Password:	<input type="password" value="....."/>
<input type="button" value="Submit"/>	

Username:	<input type="text" value="Test1"/>
Password:	<input type="password" value="....."/>
Authority:	<input type="text" value="2"/>
<input type="button" value="Register"/> <input type="button" value="Log out"/>	

Logged as: admin

Figure 4.8: Login and Registration Website

In order to achieve these results, it was created a *users* table on MySQL, which can be seen in Fig. 4.9. The table has the auto-incrementing ID for the user identification on the back-end, the user's username for front-end usage, their authority, and their corresponding hash and salt. Even though the website is quite basic, it not only has input sanitization but also protection against SQL injection, which prevents malicious attacks, ensures data integrity, protects user privacy, and can maintain an organization's reputation. By properly filtering user inputs, it is possible to deny unauthorized access to the database and avoid security breaches. Additionally, the user password is protected through a salt and hash system using the SHA-256 function. The generation of the salt is a purely random value that is appended to the password before hashing, which makes the hash more complex and unique. Then the hash is computed through its cryptographic transformation algorithm, resulting in a string that represents the password in an irreversible manner. Both the salt and the hash can be then safely stored in the database, and when the user attempts to login, the password they inserted is hashed with the same salt and then compared to the hash stored in the database; if they match, the user is validated and proceeds to the Session Scene.

<i>Users</i>	
ID (primary key)	int
Username (unique)	varchar
Authority	int
Hash	varchar
Salt	varchar

Figure 4.9: Users Table SQL

Finally, for the AR interface, the design theme aimed for was a straightforward, yet professional look, as stated in Section 4.2. Using MRTK, all the necessary inputs were placed: the user ID or username and the password. When the text input field is touched, a virtual keyboard appears in front of the person, so they can type, however, since this can be time-consuming, a bypass button was created to log in as a Guest (authority 1). The Guest login feature could be especially useful for future demos so that every person trying out the software would not need to register a new account. When pressing the *Log in* button, a PHP script is run on the HTTP server that validates the user. Also, every interactable button is compatible with not only near-touch or long-ranged pinch but also gaze and voice inputs.



Figure 4.10: Login AR Interface

## 4.4 Session Scene

After successfully implementing the user control functionalities, including the user registration via web and the application log-in interface and back-end logic, the need to provide the users the ability to create, list, and join specific online sessions arose. Initially, the project did not intend to have multiple online sessions, just one working multiuser robotic tech demo; however, the possibility of utilizing the networking library to its maximum potential in order to create an online environment capable of holding various sessions, each with different users participating, different virtual elements and different robot simulations was a very intriguing concept that could end up not only future-proofing the application but make it more compelling.

The number of robots in a specific manufacturing industry can vary significantly depending on the type of industry, the size of the infrastructures and the productions scopes, as well as the level of automation, the *International Federation of Robotics* (IFR) shared a report in 2021 [73] that stated that the robot density was averaging 113 robot units per 10,000 employees. The report also states that this value not only doubled 2015's statistics but is also expected to grow by at least 10% every year. This result is an average of all the worldwide data, where the growth in Europe and America is slower, but Asia, on the other hand, has significantly higher values, for instance, the expected expansion in Singapore is averaging 27% per year. This growth is firmly correlated to each country's economic development and labor force national rules. As the utilization of robotics expands significantly, it becomes crucial to appropriately scope and plan the multiuser-side of the project in order to handle a potentially growing number of online sessions per robot. The approach of empowering a single IP address to accommodate multiple robotic applications concurrently could significantly simplify the complexity of cloud computing or web services allocation, allowing for a dynamic evolution of the robotics landscape in the

manufacturing industry.

To address the employment of these features, it was applied a robust system that leverages a combination of the MySQL database, a REST API master server, and the networking library logic, as seen in Fig. 4.11. This figure explains in a more clear way the back-end logic overview behind the actual implementation of the concept of sessions.

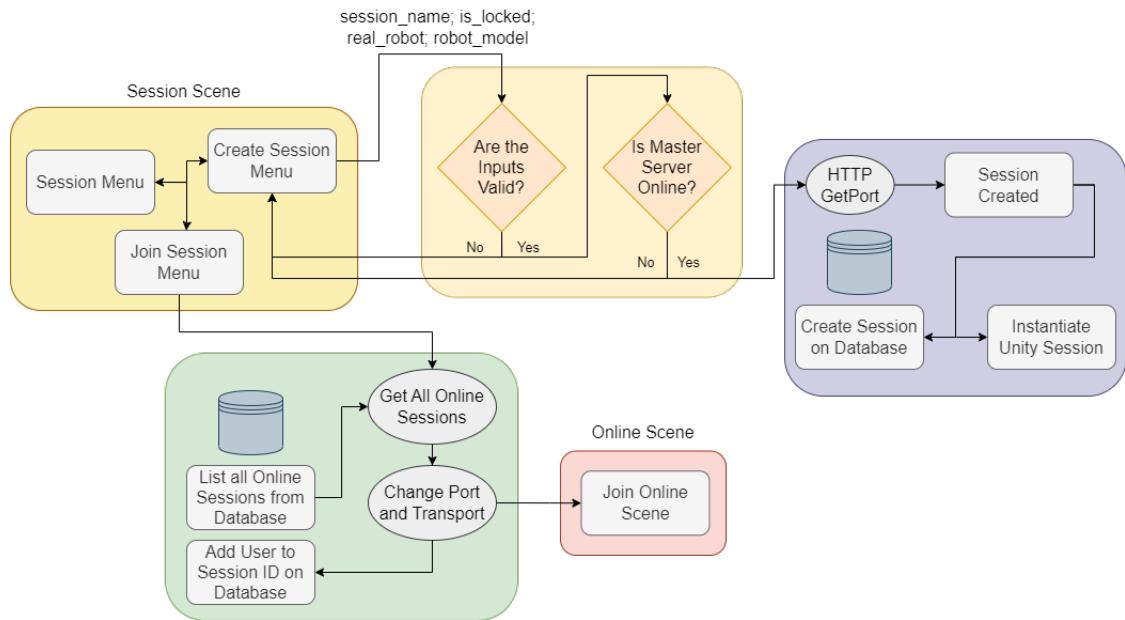


Figure 4.11: Simplified Session Scene Flow UML

Despite this part of the application still being completely offline and without actual connectivity to a server or any multiuser interaction, there was the necessity to start the deployment of Mirror multiplayer with the objective of treating the Session Scene as an offline lobby for a subsequent online session. To accomplish this, Mirror has a high-level API component that is responsible for either hosting a server or establishing the connection to the server - the Network Manager (Fig. 4.12). Mirror provides a default Network Manager that works decently as a starting point, though for the project a complete rework of the component with custom game-specific logic was required. The Network Manager is needed to set up the server IP, the transportation method and the specific port being used, the player spawn method, and the game state and scene management. Most of these features were developed later on for the Online Session, as can be seen further on in Section 4.5. For instance, the player spawn method, the user prefab, or the message send rate were concepts that only mattered when the player got inside the session, not during their lobby navigation; so, for the Session Scene alone, the main feature used for the Network Manager were the Scene Management, where the present application scene was considered an *Offline Scene* or a *Lobby*, and the initial choice for the Transport component.

The Transport component, albeit it is not supposed to be actively used on the *Offline Scene*, it is required to set up the initial configuration of the Network Manager and also to establish the

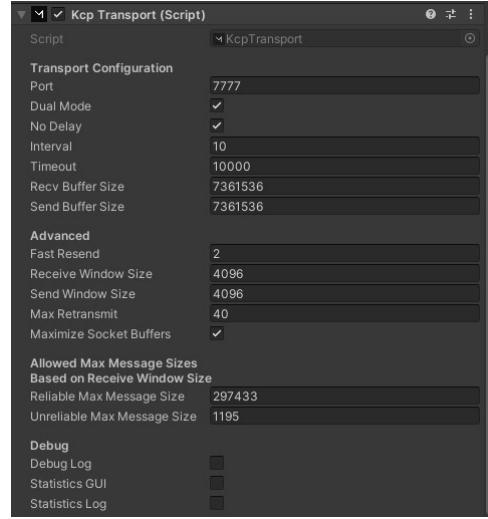
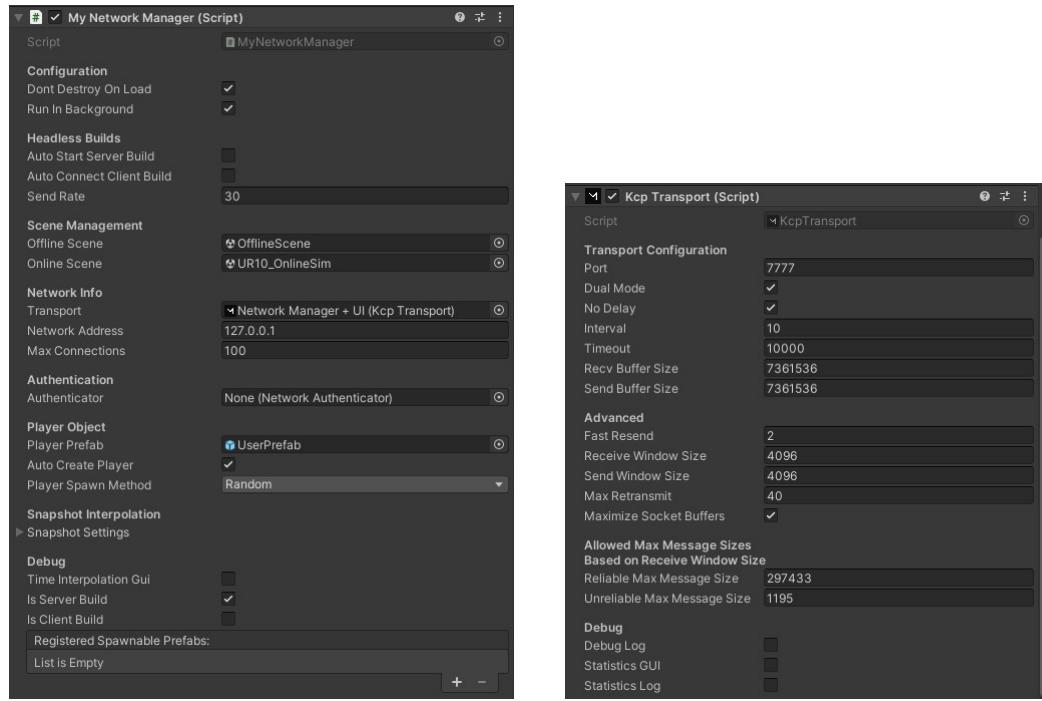


Figure 4.12: Custom Network Manager and KCP Transport Components

port that is going to be used for the *Online Scene*. For the project, two main type of transport were considered:

- **KCP Transport** - Very reliable transport layer over the User Datagram Protocol (UDP) with low latency and built-in congestion control. It offers improved packet delivery and error correction when compared to raw UDP. On the other hand, configuring and fine-tuning KCP can become a nuisance and it has limited platform support, for instance, Docker and WebGL are incompatible with KCP transport.
- **Telepathy Transport** - It is very lightweight and simple to set up, has cross-platform support across a wide range of platforms, and is especially efficient in terms of CPU and memory usage. Despite all that, it lacks more advanced features that are present on other transports, such as congestion control and is only suitable to smaller-scale projects.

Mirror also has a third featured transport, the WebSockets transport, however, as the name suggests, it is designed and optimized to accommodate modern web-based applications, even if it works decently on 2D and 3D applications. Based on these preliminary findings and after an empirical and technical approach, the KCP Transport ended up being chosen as the main network transport, although, in a later stage of the project, compatibility with all three transports was manually programmed to future-proof the system.

So that the Transport component could dynamically get an open port to use as the session's communication channel, a dedicated master server was deployed. By following a RESTful API

architecture style, an HTTP server was developed using Python in order to oversee the availability of the IP address and efficiently allocate the ports when requested. The server can work on the localhost (in this case, it was being run locally on port 10001) or can be used as a cloud solution by setting up a cloud service capable of not only running Unity instances but also with a custom firewall that allows the usage of all ports. The first main feature of the master server was the *getPort*: the user sets the start port, which is the first port that can be used as a session, and decides what the maximum number of Unity instances can be. Table 4.1 shows every port being used in the project in the localhost IP address.

Table 4.1: System's Port Allocation

Ports	Functions
4443	Apache SSL
8080	Apache HTTP and PHPMyAdmin
3306	MySQL
10000	ROS TCP Connection
10001	Master Server
10002	Session 1
...	...
10002+N	Session N+1

After receiving a *getPort* request, the master server returns an available port as plain text to the client that made the request and starts an instance of the Unity server with the specific port as an input argument at launch. When the application launches, by manipulating the plugin and asset initialization order on the Unity editor and creating a script that works asynchronously, the correct port is set in the Transport component before the Network Manager is able to start up the server. This is especially relevant to note, the Network Manager can change the transport port number mid-execution but there needs to be some kind of logic that allows the server to stop the Network Manager and restart, but since these instances were meant to be run independently and without any kind of maintenance, by executing the process in the correct order there would not be a need to mitigate possible network bugs. Fig. 4.13 shows the creation of two sessions on ports 10002 and 10003.

```
Running HTTP Server on port 10001
127.0.0.1 - - [19/Jun/2023 23:04:22] "GET /getPort HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2023 23:04:26] "GET /update?port=0&players=0 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{0: 0}
127.0.0.1 - - [19/Jun/2023 23:04:36] "GET /update?port=10002&players=0 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{0: 0, 10002: 0}
127.0.0.1 - - [19/Jun/2023 23:04:41] "GET /getPort HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2023 23:04:42] "GET /update?port=0&players=0 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{0: 0, 10002: 0}
127.0.0.1 - - [19/Jun/2023 23:04:42] "GET /update?port=10002&players=0 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{0: 0, 10002: 0}
127.0.0.1 - - [19/Jun/2023 23:04:46] "GET /update?port=10003&players=0 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{0: 0, 10002: 0, 10003: 0}
```

Figure 4.13: Creation of two Sessions

Additionally, two more features were added to the master server: the *update* call that writes on the console the port value and the number of players currently in that session, and the *status* request that generates a JSON encoded dump with all the data on the specific Unity instance. The *update* call would later be used as a way to keep track of a specific session is still in use or if the application stopped unexpectedly and the port is now free.

```
127.0.0.1 - - [19/Jun/2023 23:07:38] "GET /update?port=10005&players=0 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{:0: 0, 10002: 0, 10003: 0, 10004: 0, 10005: 0}
127.0.0.1 - - [19/Jun/2023 23:07:39] "GET /update?port=10004&players=0 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{:0: 0, 10002: 0, 10003: 0, 10004: 0, 10005: 0}
127.0.0.1 - - [19/Jun/2023 23:07:40] "GET /update?port=10004&players=1 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{:0: 0, 10002: 0, 10003: 0, 10004: 1, 10005: 0}
127.0.0.1 - - [19/Jun/2023 23:07:42] "GET /update?port=10003&players=0 HTTP/1.1" 200 -
127.0.0.1 127.0.0.1
{:0: 0, 10002: 0, 10003: 0, 10004: 1, 10005: 0}
```

Figure 4.14: Player joins the Session with Port 10004

After implementing a way of dynamically generating Unity server instances, since it was a REST API-based server, it still needed a way of keeping track of all the session data. To achieve an efficient way of controlling not only the sessions but also the user information, the database system was revised and expanded, and new back-end logic and scripting were implemented. By providing a user-friendly interface enabling the creation, listing, and joining of specific online sessions, it was possible to connect the database and master server back-end components to the application interface front-end. The new and adjusted database entity relationship diagram can be seen in Fig. 4.15. Besides the *Users* table, the new *Session List* table is used to keep track of all the data of every session currently online, and the *Sessions* table is simply a way of overseeing every user individually in a session.

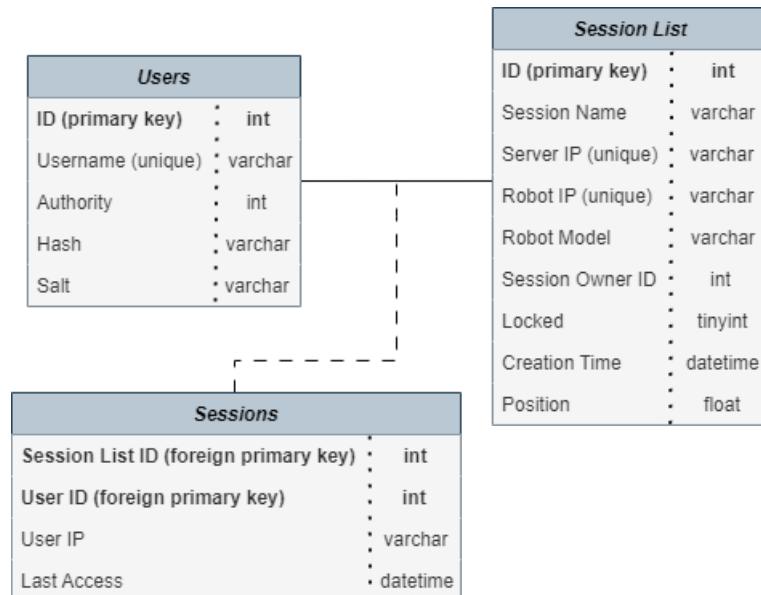


Figure 4.15: Database Entity Relationship Diagram

On the *Session List* table, it is possible to store the name of the session, the session IP address, which should be unique otherwise something went wrong during the creation of the session, the robot IP address and the robot model, if needed. The session owner ID is a foreign key to the *Users* table with the initial aim of giving the user that created the session maximum authority over everybody else in the session; the locked bool was a feature that ended up being deprecated due to not working correctly, however, the primary idea was to lock the online session to outside persons and when a user asks to join the session, the session owner receives a prompt to allow or deny the entrance of said user. The creation time field is used to control how long has the session been online and the position value was an early test for positioning the hologram depending on the marker location, as in, if someone moved the 3D model from its initial positioning on the marker, when an outside user joins the session the hologram would synchronize to the new position automatically. The data instances inserted on the *Session List* and *Sessions* are temporary and updated as needed: when a user leaves the online session or if the session is stopped, all the corresponding data is either revised or removed from the database.

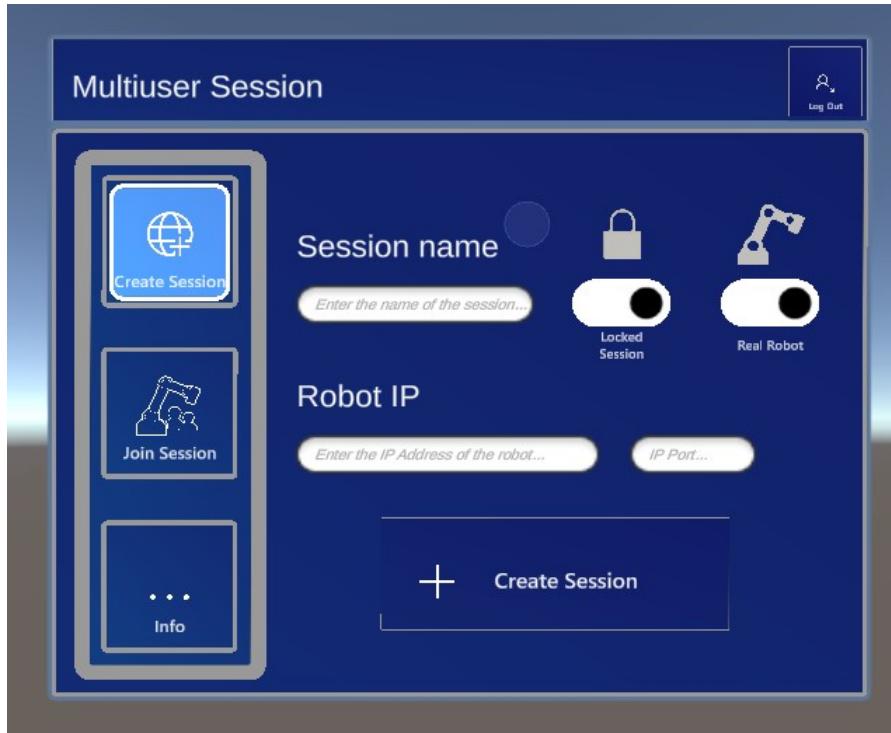


Figure 4.16: Create Session AR Interface

Fig. 4.16 shows the UI when the user is creating a new online session. It has all the valid inputs necessary to generate the session, such as the session name and the robot IP, although the session name can be left blank for a default session title. The *Real Robot* button is a toggle switch, and when it is off the robot IP input disappears, otherwise, the robot IP has to be valid to proceed to the port number acquisition. When pressing the *Create Session* button, it sends the *getPort* request, receives the port number allocated to the new session as plain text, and while the Unity instance is being launched, a PHP script updates the *Session List* table with the new session data.

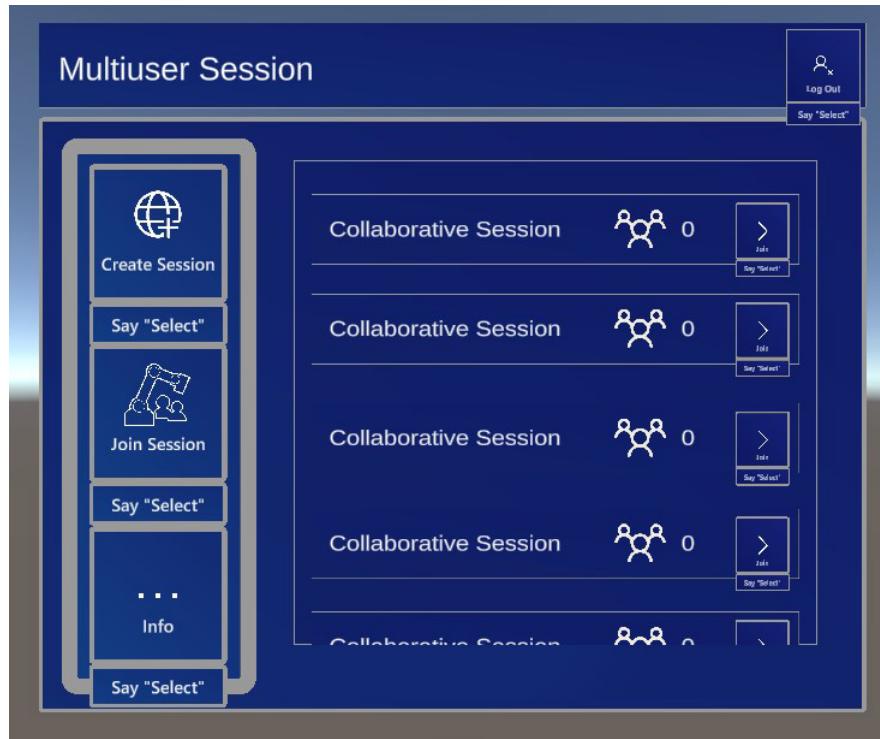


Figure 4.17: List and Join Sessions AR Interface

Finally, Fig. 4.17 depicts the interface that lists every online session currently available by fetching each element on the *Session List* table and instantiating a dynamic GameObject for every session. The user is capable of scrolling through the list by near-touching it or using the pinch gesture (near or far interactions); when the user finds the session that they want to join, they just need to press the *Join* button. By keeping the interface organized and avoiding cluttering the UI with visual or interactable objects, whilst still trying to keep a professional outlook, the final result is a dependable, user-friendly UI that is capable of blending in with the real world while being virtually recognizable, enhancing the user experience as a result.

## 4.5 Online Scene

For this part of the system and for the ROS Connection, it was used a tech demo previously developed at INESC TEC. Initially, the demo used the library *ROS# (ROS Sharp)* and *ROS Bridge*<sup>7</sup> in order to communicate with the robot through the Unity engine. Even though it was already capable of fully controlling real robots, simulating environments and programming C# commands, while also supporting publishing and subscribing to ROS topics, the Unity Robotics platform is not only more straightforward and simple to use when compared to the *ROS#* library, but it also offers a higher level of available tools for robotic applications while also relying upon an active online community that is continually creating new and original open-sourced features every day.

<sup>7</sup> <https://github.com/siemens/ros-sharp> and <https://github.com/EricVoll/ros-sharp> for some UWP showcases (e.g. for the HoloLens)

Therefore, a new and upgraded demo was made by INESC TEC using Unity Robotics and its vast library of assets and plugins that replaced the previous *ROS#* demo.

Albeit the dissertation's project has a tight relation with robotics, in the end, by having a feature-rich and working tech demo available, alongside a very knowledgeable and experienced team, there was the possibility of adapting the dissertation project in order to accommodate the already-existent ROS architecture. Still, multiple points should be taken into account when integrating the ROS demo: The project's new technologies have to be compatible with all the APIs and libraries being used in the ROS demo in order to avoid conflicts; the project's communication protocol needs to be compatible with the ROS TCP connection, to avoid the need to appropriate and adapt messaging formats for data exchange; the network configuration of the project has to be adapted so that the required ports and firewalls are properly configured and compatible with the ROS demo; also, the existing hardware and infrastructure needs to support the new system to avoid system crashes or performance bottlenecks. The architecture analyzed in Section 3.3 was already planned for these possible conflicts, which means that, overall, the project's integration of the ROS demo is expected to be fairly suitable and smooth.

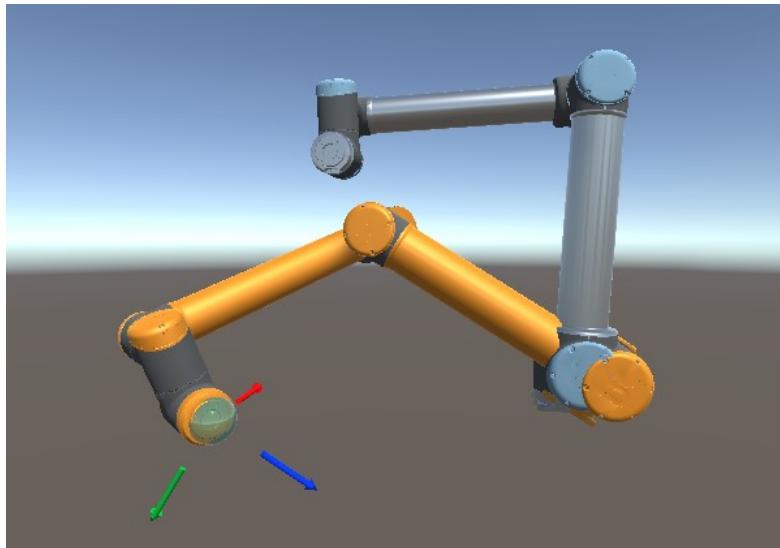


Figure 4.18: ROS Tech Demo for UR10

Fig. 4.18 shows what is displayed to the client on the HoloLens 2, where the gray, bluish robot is the Digital Twin of the real robot, and the orange one is used to program the robot's movement. The demo runs by first manually inputting the robot's IP address, which could either be a real robot or a simulated one on the Virtual Machine. In order to be compatible with the new way sessions are created, since the robot's IP address is already defined as an input field in the UI (Fig. 4.16), even though this feature was not technically removed, now the ROS TCP Connection is automatically started on launch by receiving the IP address as an application execution argument. It is relevant to note that, to accommodate the ROS architecture from the tech demo, only one port was being used for the TCP connection, so only one client could be connected at a time. To solve this, the build that is connected directly to the ROS is the session server and any client could send commands

directly to the robot through the server. This is accomplished during the startup of the application, ensuing the connection of the server with the robot, the ROS TCP Connection GameObject is destroyed on every client's end and their programming commands are instead deflected to the session server that, consequently, delivers the calls to the ROS (Fig. 4.19). By redirecting the command through the server, it is possible to achieve several advantages over a direct connection to the ROS, for instance:

- Security - By routing the commands through the server, it is possible to implement security measures to protect the end destination, which could potentially avoid disastrous events to the hardware.
- Control - By implementing policies, monitoring, and auditing mechanisms on the server, it is possible to ensure compliance and track command activity. Additionally, using the server as a logic buffer, it is possible that higher authority users override commands from regular users.
- Scalability - The server can act as a load balancer for the ROS server, since before multiple users would have a direct connection to the robot. This optimizes resource usage and facilitates scalability, as it is possible to add or remove features without hardly affecting the end-client.
- Optimization - Before, the computation of the commands was all done on the HoloLens 2, which already has limited resources, but by using the server to compute potentially hard commands on a computer, it is possible to reduce the workload on the client, improve response times and reduce network congestion.
- Monitoring - Redirecting commands through the server provides an opportunity to create logs and monitor all request traffic, which can be used to analyze and troubleshoot conflict by capturing metrics and generating error reports.
- Simplification - Using the server as a medium provides an abstraction layer, which hides the complexity of the application from the user.

Although redirecting commands through the session server was the considered route planned for the expansion of the ROS demo, most of the work was put into the networking library and getting the multiuser experience working, so, a simple ROS Publish feature command was created and tested, which in the future can be used as an example for further development of the ROS programming. Additionally, this demo was decomposed into two different scenes: one with the real robot, and another with a fake manipulable robot. The latter scene was conceived with the idea of debugging the multiplayer features since it was significantly faster to test and build, plus there was no need to use the VM, but it was also designed to use for training and demonstration purposes. The scene was defined as the *Online Scene* on the Network Manager component, so, when a user clicks on joining a session, this scene starts up automatically and every user can interact with the fake, manipulable robot.

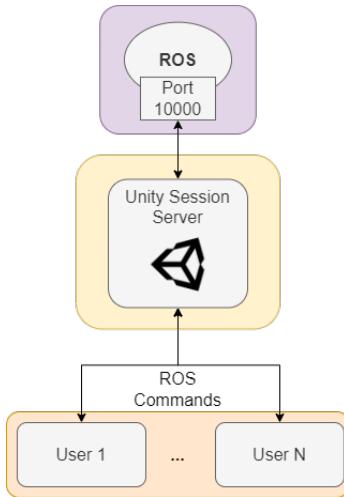


Figure 4.19: Users’ ROS Commands are routed through the Session Server

To achieve data and game state synchronization across every user, as in, every participant of the session should be visualizing the robot moving around in real-time, other Mirror components were used - the Network Identity and Network Transform. The Network Identity component controls a game object’s identity inside the network and uses that identity to make the networking system across every user aware of that Game Object; the Network Transform component synchronizes the position, rotation and scale of a networked Game Object. By empirically testing these components and after debugging system incompatibilities, it was possible to achieve data synchronization across every user.

To manipulate the robot, the user grabs a small sphere, known as *Target* (Fig. 4.20), and moves it to the desired position. Later on, in order to optimize the data transmission and bandwidth, instead of synchronizing the whole robot, only the Target was synchronized across the network, and since the robot’s positioning calculation algorithm was the same on every client, the robot should have in theory its position correctly set for everybody. After implementing this new method, the data usage was significantly improved since the robot was a very complex body imported from a URDF XML-based file. Initially, the server was sending on average 62 messages of 213,23 KBytes per user, and after optimization, the server was instead sending 30 messages of 310 Bytes per user, which is a considerable upgrade since this would scale exponentially with each new user joining the session.

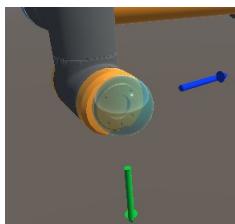


Figure 4.20: Robot’s Programming Target

However, there was still another issue related to the manipulation of the robot, for instance, two users shouldn't be able to manipulate the robot at the same time or else there would be network conflicts or the robot Target wouldn't update correctly for everybody. To solve this, it was added an authority requirement to use the Target: if a user grabbed the Target, they would instantly remove the authority from the previous person that moved the robot, and this user now had complete authority over the object. This had to be programmed carefully in order to work without any concern since the Target manipulation was the main form of network interaction in the session between users. Other networked components were added to enrich the multiuser experience, such as the synchronizable workspace (Fig. 4.21). The robot workspace was developed at INESC TEC and it was designed to use for programming by demonstration [63], however, for this tech demo it was used more as a visual cue to show the space where it is possible to physically collaborate with the real robot. If any user presses the *Workspace* button on the interface, the robot's workspace would show up for everybody in the session, independent of their authority.

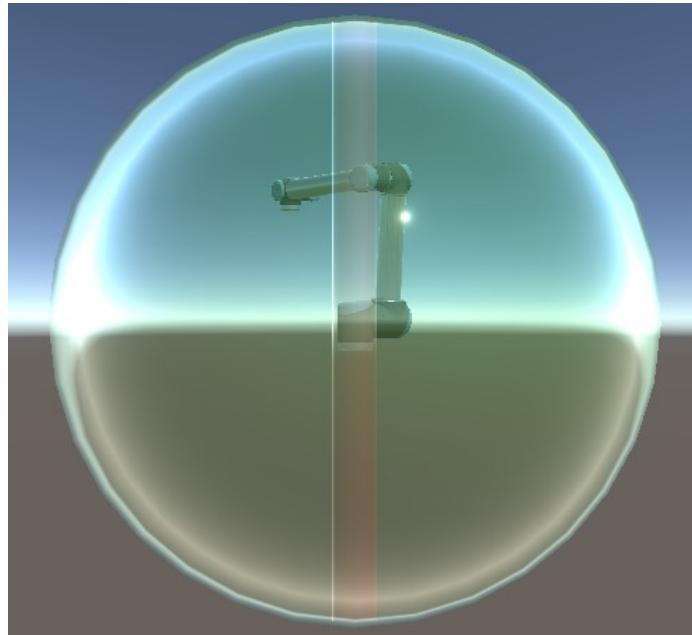


Figure 4.21: Robot's Workspace

To facilitate the acknowledgment of who was participating in the session, a player prefab was conceived, as seen in Fig 4.22. After some scripting and back-end configuration, a simple green sphere would appear above anybody currently in the session, allowing for a more fluid experience. Although, this generated another problem, since in AR it is complicated to know the absolute position of a person without using markerless algorithms. So two solutions were experimented with: one using a marker to triangulate the user's position and another using the HoloLens 2 starting position.

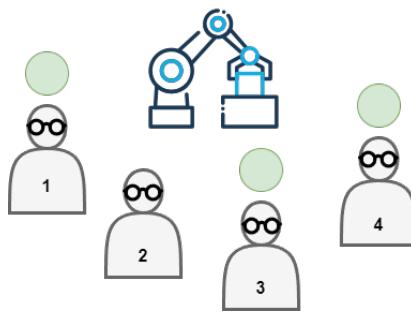


Figure 4.22: Player Prefab above Users 1, 3 and 4 Heads

In order to implement marker-based positioning, it was used Vuforia, a platform that allows developers to create and deploy AR experiences through the use of tracking real-world objects, like markers. Additionally, Vuforia made available recently new demos especially developed for the HoloLens 2. The marker recognition, in this case, a QR code, was contrived with the use of the Image Target Behaviour component (Fig. 4.23); the models that should appear in relation to the marker were just GameObjects children of the parent containing the Image Target component.



Figure 4.23: Robot Model appearing over QR Marker

Though Vuforia made marker-based application development especially accessible, in the end, it not only made the debugging process very difficult and time-consuming, since the use of the web camera was no longer possible after setting up the Vuforia Play Mode for the HoloLens 2, but the user also had to continuously observe the QR code to visualize the robot, which could detract from the user experience by limiting the viewing space and interactivity. One of the main points of AR is the immersive experience, and hindering the user by requiring a constant focus on the target limits the versatility of an AR application. Vuforia also makes use of licensing and activation keys, which arises concerns about further monetizing and the project's long-term viability and budget resize, for instance, the target models are only considered for prototyping usage only on the free subscription; furthermore, the need for an external database for image recognition not only adds complexity to the project but can potentially create dependencies as it introduces the need to rely on external services. These drawbacks and unfavorable outcomes ended up resulting in the total

removal of the marker detection feature using Vuforia for the moment. Since other colleagues at iiLab are researching marker-based detection algorithms and experimenting with other libraries, in the near future the project could once again incorporate markers for hologram positioning, but in the meantime, the positioning of the hologram was made depending on the HoloLens 2 starting position.

Although the removal of the marker-positioning feature could be seen as an inconvenience, by not relying on an absolute physical positioning mechanism the system's actual architecture can conveniently incorporate remote multiuser AR experiences. Considering the session is allocated on a device inside a Local Area Network (LAN) which every participating user connects to, if there is a VPN router or a computer running a VPN software like OpenVPN<sup>8</sup> and enabling port forwarding inside the LAN it is possible for a remote user to connect to the sessions via VPN and join the multiuser experience. In the future, if this feature becomes a priority, there could be simpler ways of implementing this, by allocating the sessions and master servers in a cloud server with a dynamic DNS, for example. On the other hand, this would also raise some new conflicts, especially considering that remote users could not communicate so intuitively with other participants inside the session: this could be solved by incorporating a microphone or voice recording processes or a more complex system like turning each participant's hand into a networked object that everybody could see inside the session - MRTK tracks the user's hand for reading gesture inputs and also creates a virtual hand model that mimics the real hand movements.

Finally, having a completely independent UI in the AR multiuser experience is crucial for a seamless and immersive experience. In such scenarios where multiple individuals are simultaneously interacting with the AR virtual elements, an independent UI becomes necessary for ensuring each specific user has a personalized interaction based on their own unique perspectives. This also allows each participant in the session to explore and interact with the virtual elements in a manner that suits their need, and since this independent UI is only shown to their owner, it allows for collaboration without hindering other users' views. To carry out this objective, the UI was programmed so that it becomes active only on each user's specific camera.

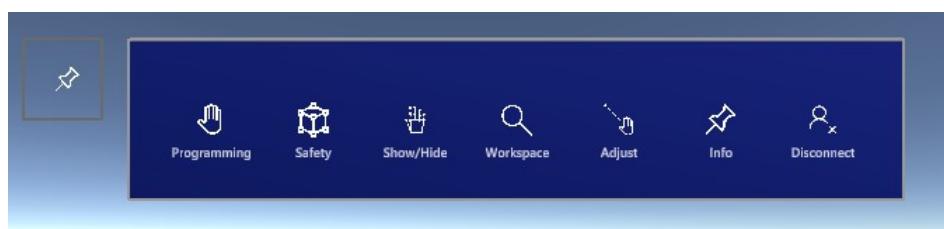


Figure 4.24: Independent Session UI

The UI, as observed in Fig. 4.24, is composed of mostly the original UI from the ROS tech demo, but with the networking components added. There is a *Pin* button to stop the UI from following the user camera around, the *Programming* button that in the real robot scene activates the manipulable robot which the user can move around to program and send the position to the ROS,

---

<sup>8</sup><https://openvpn.net/>

but in the simulated robot scene it allows to reset the networked robot position. The *Show/Hide* button allows to toggle the robot visibility and is completely independent of other users' views. The *Workspace* button enables a sphere around the robot and, as a networked object, it is synchronized and enabled to every person in the session; on the other hand, the *Safety* button is a feature slightly related to the *Workspace*, as in, it also generates a field surrounding the robot that when entered it slows down the robot's movement speed, but this feature was left as independent on purpose. The *Adjust* button allows to move around the robot; initially, this new position was going to be stored on the *Session List* database so that when new clients join the session they can see the robot on the updated position in relation to the marker, but since the marker detection feature was removed, it lets users change the robot's position for everybody but with no absolute fixed positioning in relation to the real world. The *Info* button is a feature on the ROS demo that shows relevant data on each joint of the robot; and, finally, the *Disconnect* button, that lets the user freely leave the session and reload the *Offline Scene*, where they can choose another session to join. When this button is pressed, the *Sessions* database table is updated and if there is no user left on the session, the application is closed and the session is terminated on the *Session List* table. Additionally, it was added a HeartBeat script to the application to communicate with the master server every five seconds, updating it with the number of users and if the port is currently being used; when the HeartBeat stops, the master server recognizes that the application probably terminated unexpectedly and frees up the port that was being used.

## 4.6 Application Deployment

Since all of the development and testing was completely made on the computer environment and emulated on both the Unity Player and HoloLens2 Emulator, the application was bound to have some problems when deploying. Most of the issues were related to the build method for the HoloLens 2 platform (UWP), although, other issues also stood out during physical testing. For instance, the *Sessions* table was not updating correctly when on the HoloLens 2, for an unknown reason, and the master server could only pass a single argument to the application launch, which meant that the robot IP being appended to the launch argument had to be removed from the master server. The HoloLens 2 application FPS was extremely low, but this was corrected when building in a lower-quality graphic resolution. Finally, the networked objects were not working as intended and sometimes the objects would not synchronize, this was due to the HoloLens 2 input system being very specific and not compatible with some of the scripts created. After some tweaks and adjustments, the application was running smoothly, being capable of creating and joining sessions, and interacting in a multiuser AR environment. As a side note, it is possible to check that the user position synchronization was working accordingly on Fig. 4.29, where the green sphere that indicates that a user is inside the session is above both a user's head but also above the computer that was also a client participating in the session. Fig 4.28 shows how manipulating the robot on the HoloLens 2 also updates its positioning on the session server running on the computer.

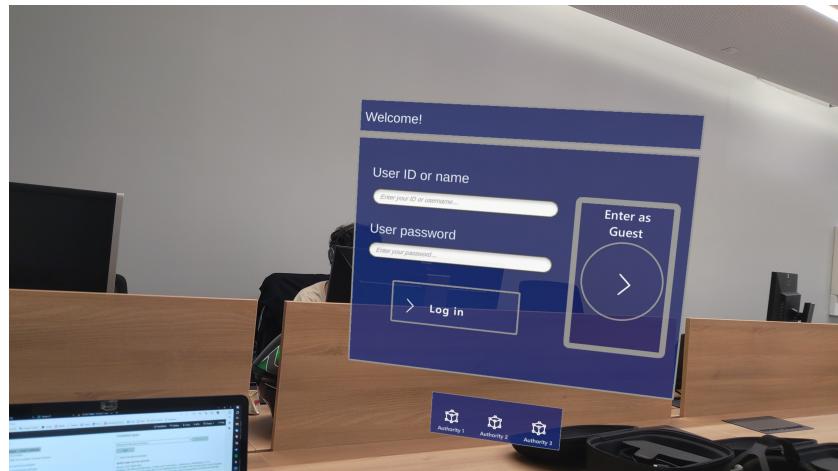


Figure 4.25: Login AR UI on HoloLens 2

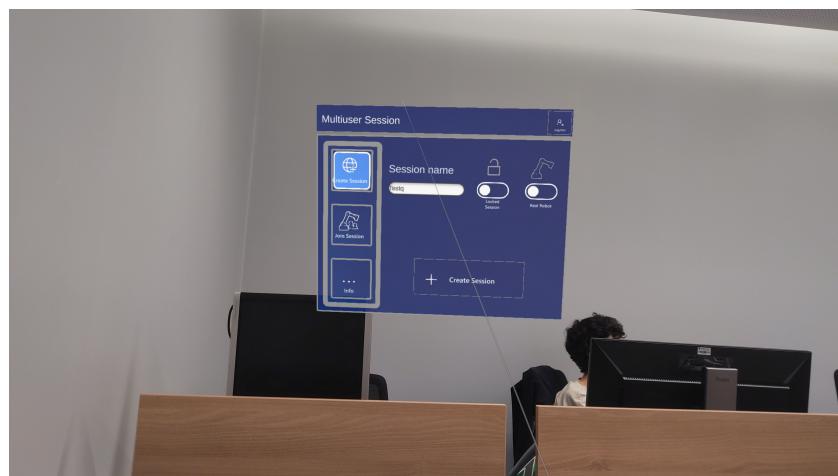


Figure 4.26: Session Creation AR UI on HoloLens 2

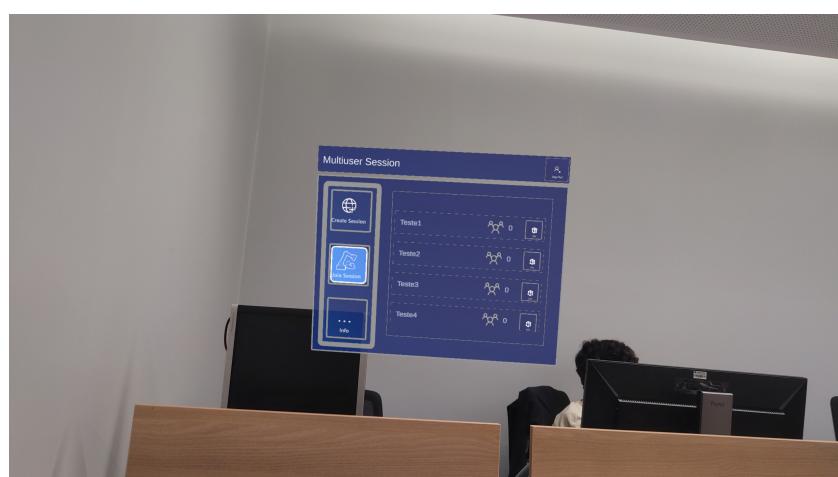


Figure 4.27: Joining Session AR UI on HoloLens 2



Figure 4.28: Manipulable Robot Synchronizing both on Client and Server's ends

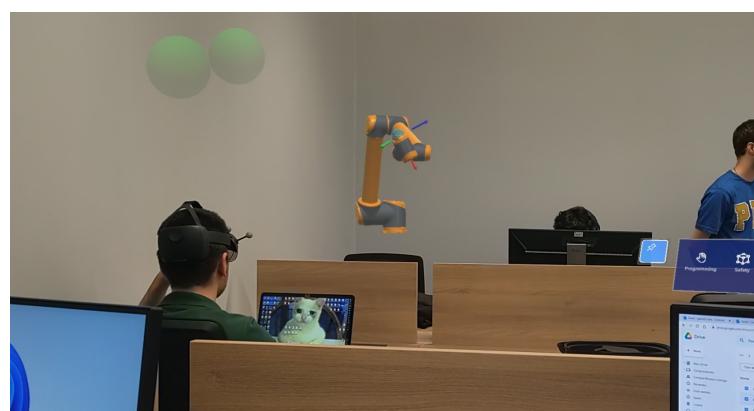


Figure 4.29: Manipulable Robot during Session and User Prefabs above Participant's Heads



Figure 4.30: Session Control AR UI on HoloLens 2

# **Chapter 5**

## **System Evaluation**

Evaluating the final result of the system is a critical process to ensure that the end-product software is reliable and efficient, and that also achieved all of the objectives the project proposed. This chapter focuses on the analysis of the system utilizing essential tools and methodologies to assess its performance, capabilities, and user experience. These results are vital in order to identify areas that lack optimization and are in need of improvement. These assessments are conducted as experimental research, specifically focused on stress-testing the online session system in order to analyze its performance under extreme conditions, but also the gathering and interpretation of users' feedback on the final UI/UX.

The aim of subjecting the online system to stress tests is to examine its stability, responsiveness, and scalability when handling high volumes of Concurrent Users (CCU). By employing Python programs for automatizing the testing, Unity scripts to output results and Excel for value analysis and organization, it is possible to simulate heavy traffic scenarios and identify critical thresholds, points of failure and areas requiring optimization. It should be emphasized that these results depend significantly on the device's hardware specifications used to operate the system, so the data acquired is not absolute and should be instead taken as auxiliary information to evaluate the project. The target device specifications are:

- Windows 10 x64-based Laptop
- Integrated GPU - Intel HD Graphics 630
- Dedicated GPU - NVIDIA GeForce GTX 1050 (Mobile)
- CPU - Intel Core i7-7700HQ, 2.80GHz
- RAM - 16 GB

Furthermore, the value of user feedback on the UI is an essential component for the system evaluation. By presenting a web single-player version of the application to online participants, they were invited to test its functionalities and share their opinions on a feedback form. Through

this user-centric evaluation, it is possible to gain insight into the application's usability, user experience, and areas of improvement. The data collected could also guide future UI design iterations, aligning the system with the user expectations and requirements.

## 5.1 Server-Client System Stress Test

### 5.1.1 Testing Tools

The tools used to test the capabilities of the sessions were mainly composed of Unity scripts to output data collected inside the application, such as the number of messages sent between the server and client or the data usage; Python scripts to automatize the testing, making the data acquired more reliable, and allowing for easy repetition on the tests performed; and Excel for charting and organizing the data. Additionally, it was used Unity Profiler<sup>1</sup> to get information on the application's performance in areas such as CPU and memory usage.

Mirror itself was used to calculate the Round Trip Time (RTT), due to the time synchronization feature present on the network library, allowing the clock on each client to be adjusted depending on the server's clock. The accuracy of the RTT calculated is not definitive, with the possibility of having some kind of standard deviation time, which should be taken into account in the final RTT values. Additionally, Wireshark<sup>2</sup> is also effective at analyzing latency, packet loss, or other issues by capturing and recording packets and the network traffic. Both these tools were combined to check if there was any significant difference in the results between them, and in the end, by empirical experimentation, through the use of the Wireshark, it was possible to confirm that the RTT values from Mirror were very accurate.

Finally, it was also used an open-sourced software to simulate poor connection or network congestion issues in order to catch problems related to a possible broken network system: Clumsy<sup>3</sup>. Clumsy captures the network packets on the device and replicates network problems like lag by delaying their redirection; it can also generate packet-drop by tampering with the network data received; deliver packets out of order or even create a network throttle by queuing up a bunch of packets in a row. Even though this software is more useful when working with Low-Level API (LLAPI) network libraries, it is still convenient to check if there is any unexpected problem or a broken network. Before performing these stress tests, it was necessary to assure that the network could handle all these extreme case scenarios, so after experimenting for a while with Clumsy, it was possible to verify that, even though the network conditions were significantly worse, the session system still worked as expected.

### 5.1.2 CPU and GPU Usage Tests

The first obvious objective of these tests is knowing how many instances of the Unity application the targeted device could simultaneously execute by testing its limits. Each application launch was

---

<sup>1</sup><https://docs.unity3d.com/Manual/Profiler.html>

<sup>2</sup><https://www.wireshark.org/>

<sup>3</sup><https://github.com/jagt/clumsy>

scripted to happen every five seconds, although the actual start-up time of each application would exponentially increase depending on the number of instances already opened. This delaying effect was a result of the fast CPU and GPU usage generated by each instance individually. It is relevant to note that, even though these components' tests were performed in a clean system, there were still background processes and system services running while the attainment of the data.

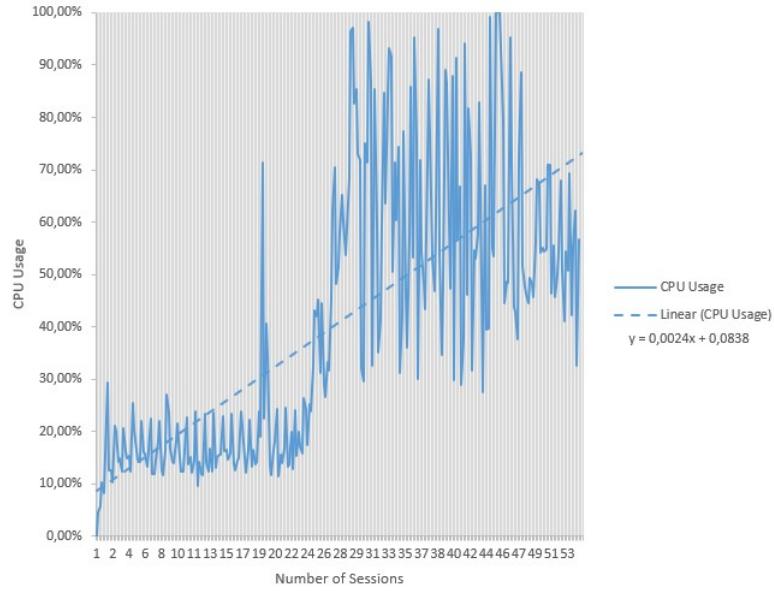


Figure 5.1: CPU Usage per Session Instance

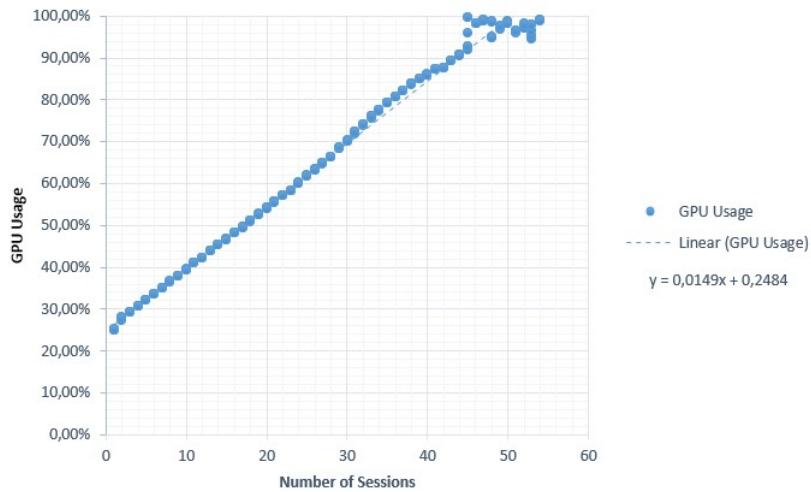


Figure 5.2: GPU Usage per Session Instance

The charts seen in Fig. 5.1 and Fig. 5.2 test how many sessions the hardware can withstand. Although the initial CPU usage is not deeply affected, after reaching 25 sessions it is possible to visualize that the CPU usage raises significantly until the end of the data gathering. The initial low CPU utilization (below the 30% value) is justified due to the CPU being capable of handling

these instances reasonably easily and also because the tasks performed by the instances are parallelizable, but when the resource competition starts, the CPU consumption increases rapidly. Every single instance requires a portion of CPU and memory, and when any of these resources start becoming limited, it causes contention that leads to the rapid breakdown of the CPU management. On the GPU side, it is possible to examine the linearity of how much GPU consumes each instance, averaging 1,49% per session launched. The script was programmed to test the maximum number of instances until the device would unexpectedly stop the script, and the maximum number of sessions is 54. But since it caused a memory leakage-related Blue Screen of Death on the device, to avoid harming the hardware, the maximum number was reduced to 43 where the GPU consumption is still following linearity and does not reach 90%. In addition, the disparity and unpredictable values seen at the end of the graph, after the 45th session, are related to memory overflow or possibly synchronization issues as well, as every instance was being run in parallel.

A similar test is run again, although this time it is focused on how many clients can connect to a specific session and what the CPU and GPU usage is. Although this experiment does not bring much more insight into the data already acquired, it is still relevant. Since every user is directly connected to the same port, even if port usage does not have a direct impact on CPU consumption, the amount of data being sent and received can indirectly affect CPU and memory usage. This happens for multiple reasons, such as:

- Network Processing - The CPU is responsible for managing network packets by routing packets and handling protocols, for instance.
- Service Load - By having a higher data traffic on a specific port, it is possible to increase the workload for the service associated with that port, in this case, the Unity client instance.

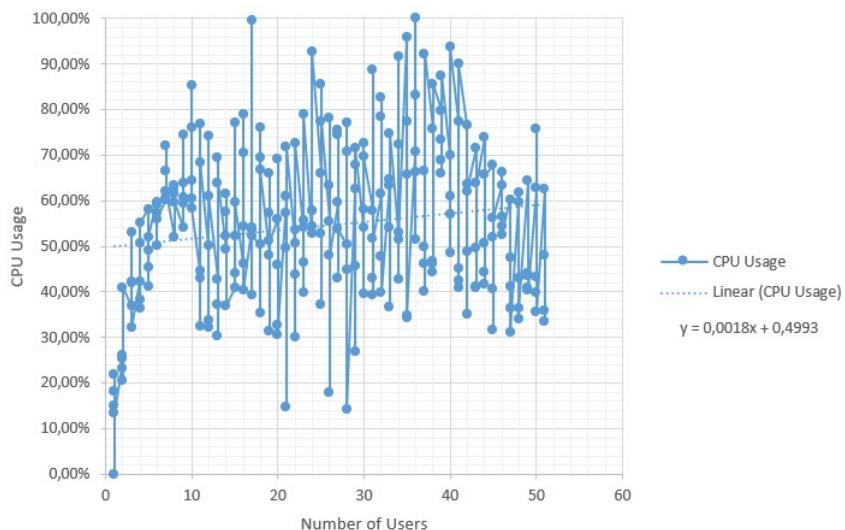


Figure 5.3: CPU Usage per Client Instance in a Session

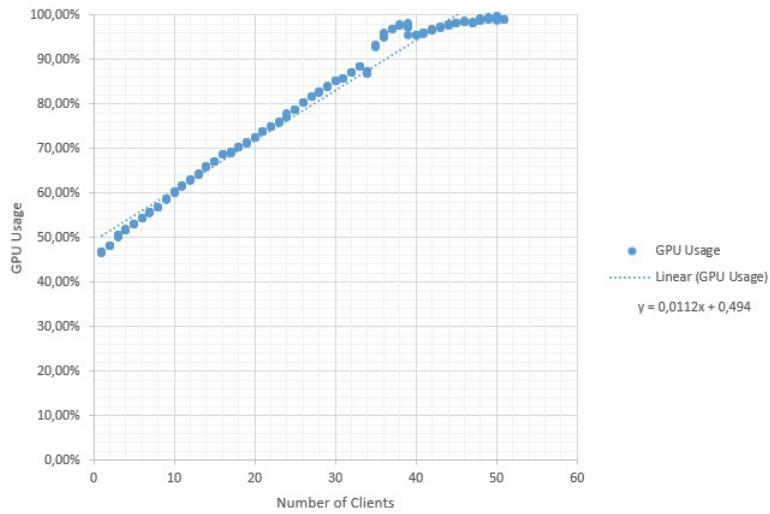


Figure 5.4: GPU Usage per Client Instance in a Session

The GPU usage-related chart (Fig. 5.4) is very similar to the server-side chart, although it has a slightly lower average GPU consumption - 1,12%. On the other hand, the CPU chart (Fig. 5.3) confirms that its usage reaches higher values even more rapidly. The high variation shown in the chart is expected to be originated from the port usage and the data that is being received or transmitted: KCP Transport includes bandwidth availability estimation algorithms and is capable of adjusting the transmission rate accordingly in almost real time. When the transport estimates that the available bandwidth is quickly decaying, it adjusts its port usage which lowers the CPU consumption, however, every new instance also uses up more CPU - this back-and-forth may generate this instability in the CPU usage chart. Finally, by applying an optimization model on Excel with the data acquired and the linear regression equations, it was possible to estimate a favorable number of sessions and clients on the targeted device: either 14 sessions with 3 users each, or 5 sessions with 8 users each; however these values do not take into account the data usage and how the network connection may decay depending on the number of users in the same session. Once again, it should be noted that these results depend widely on the testing hardware, so these conclusions should only be taken into account for the device used.

### 5.1.3 Data Usage Tests

The upcoming tests focus primarily on evaluating the data usage of the server. This is achieved by monitoring the number of packets sent and received per second, along with tracking the size of each packet in Bytes. It is possible to gain insights into the data transmission efficiency and overall network performance of the server. It is also crucial to ensure optimized data transfer rates and identify potential bottlenecks or limitations.

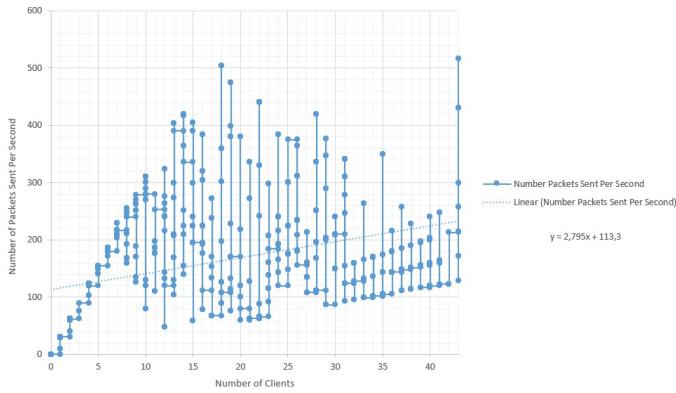


Figure 5.5: Number of Packets Sent by the Session per Second

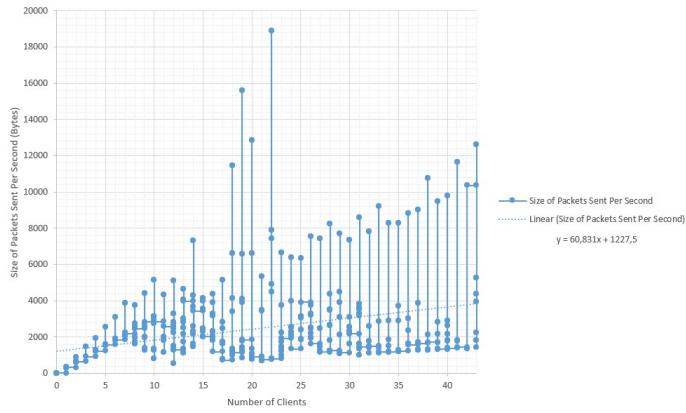


Figure 5.6: Packets Size Sent by the Session per Second

The chart seen in Fig. 5.5 refers to the number of packets that the server sends out per second to everybody inside the session while the chart depicted by Fig. 5.6 is the size of each packet sent per second in Bytes. While the number of clients connected to the session is small (until about 15 users), the number of packets sent increments per new user by, on average, 30 messages, and the size of each message is around 10 Bytes, since the server can typically handle the processing required by KCP without significant issues. However, as the number of users grows and the network traffic increases, which can lead to congestion, the server's processing capacity may bottleneck. By this stage, the KCP transport's bandwidth optimization algorithm and congestion control mechanisms start which turns the number of packets sent more unpredictable and creates outliers in the size of the messages sent. This is caused by the dynamic adjustment of the sending rate and packet size; also, it is possible to note that the KCP transport sends an irregular number of short-sized messages and overcompensates with one or two significantly sizeable packets. After reaching 30 users, the network seems to stabilize once again by increasing the number of messages sent linearly, however, the presence of a large number of small packets and a single overcompensation packet is still observed. It should be noted that even though the number of packets sent increased, the number of users in the session also keeps growing, so the raising number of mes-

sages is actually being distributed by every user, which means that in a specific user the number of packets received may be diminishing instead.

By comparing side-by-side with the charts related to the number of packets received on a specific client inside the session and their respective sizes (Fig. 5.7 and Fig. 5.8), it is possible to note that the average number of messages received is actually decreasing. Still, there is the presence of irregularities where the client receives a low number of packets and immediately after gets a larger value. The KCP transport has a rapid recovery mechanism that relies on temporarily sending out more small-sized messages to avoid network traffic bottlenecks and then, very temporarily, allows the congestion window to exceed its size and overcompensates on the number of messages sent. The aim of this technique is to recover from the congestion faster while also trying to maintain high throughput. Finally, it is possible to note that not only the number of messages is decreasing per new user introduced to the session but also the correspondent size of the overall packets received - with some outliers that are explained as overcompensation packets.

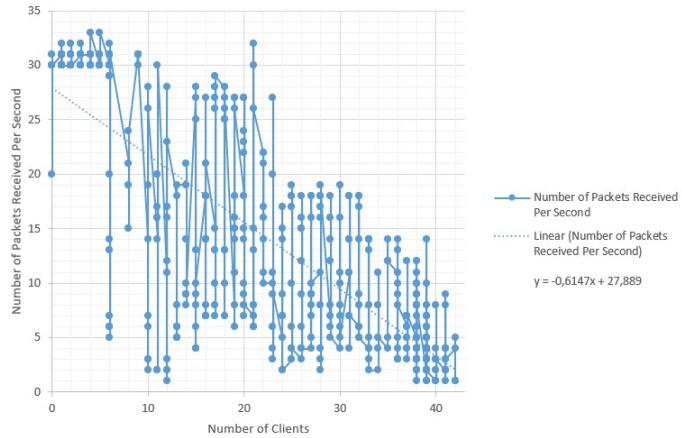


Figure 5.7: Number of Packets Received by the Client in the Session per Second

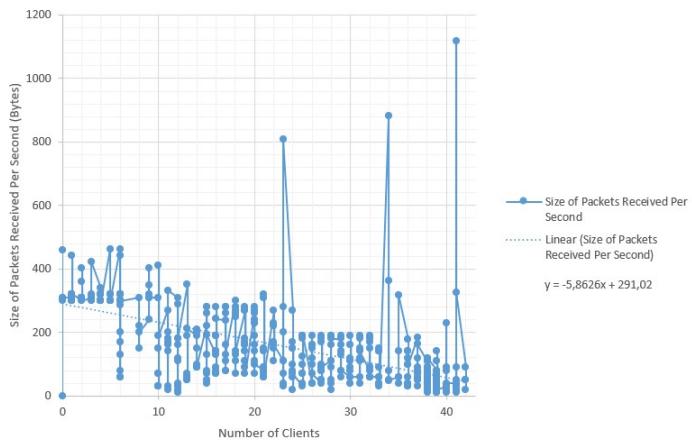


Figure 5.8: Packets Size Received by the Client in the Session per Second

The charts seen in Fig. 5.9 and Fig. 5.10 depict, once again, an analysis on the server-side but this time the number and size of packets received instead. The first thing to note is that these graphs are almost identical - this is because there is almost no packet-drop on the localhost and each message has an average of 14,22 Bytes and their size is very constant, with very slight fluctuations. Furthermore, until the first 15 users, the number of packets received is continuous, but then it becomes more unpredictable since it starts taking advantage of the KCP algorithm mentioned previously which attempts to prevent network congestion. By analyzing in conjunction with the charts depicted in Fig. 5.11 and Fig 5.12, where the sent-rate and message sent size are evaluated in a specific user, it is possible to conclude that for the first 15 users, the algorithm attempts to deliver every full-sized message, but as the network starts getting more connections, the optimization algorithm allows the client to send messages only when necessary while also relying on the rapid recovery mechanism.

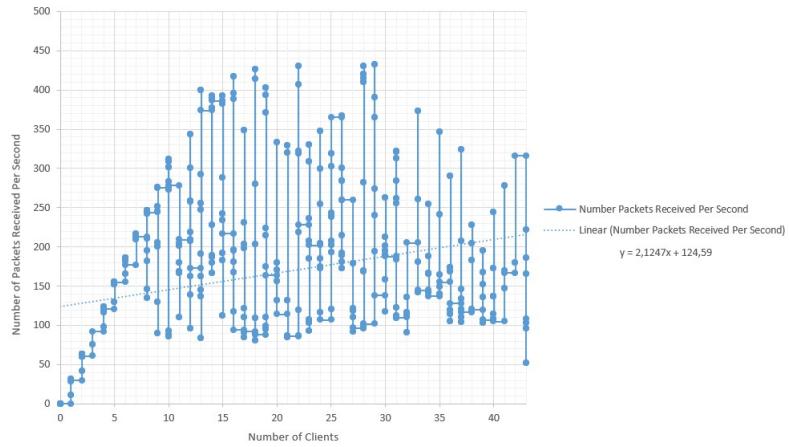


Figure 5.9: Number of Packets Received by the Session per Second

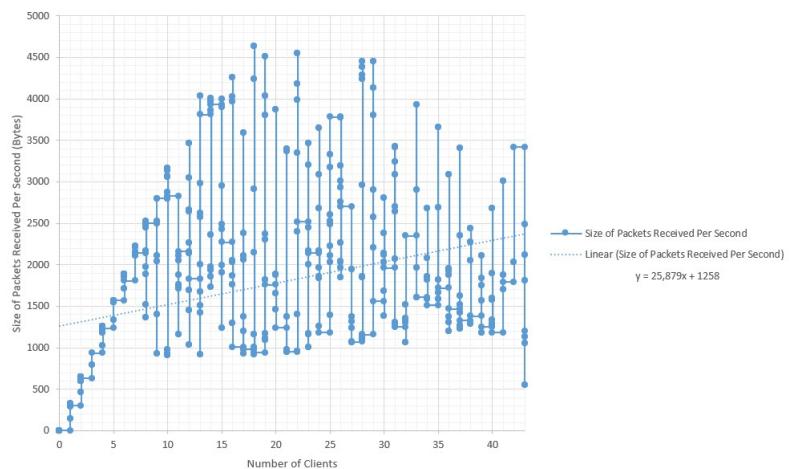


Figure 5.10: Packets Size Received by the Session per Second

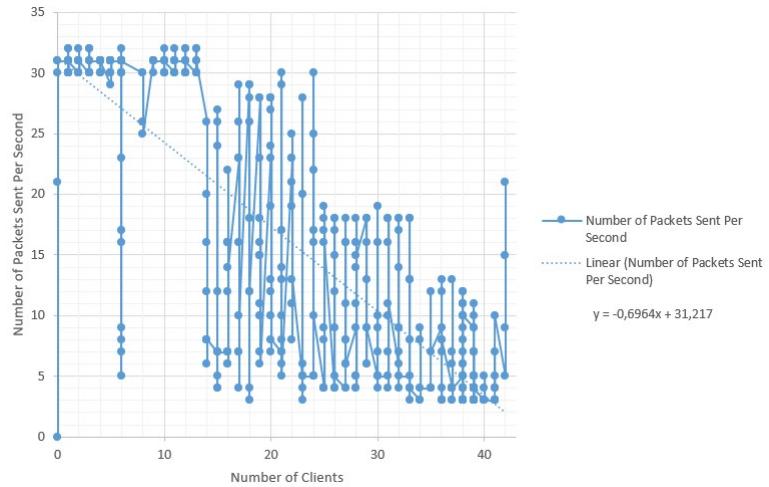


Figure 5.11: Number of Packets Sent by the Client in the Session per Second

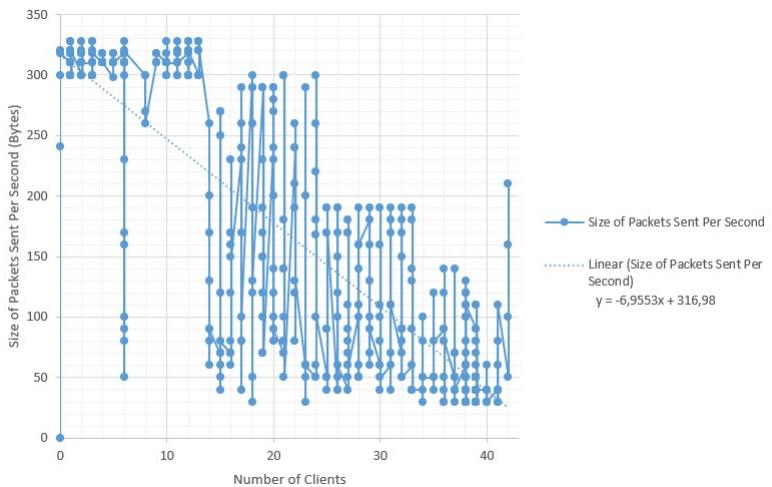


Figure 5.12: Packets Size Sent by the Client in the Session per Second

By taking these results into consideration in the number of users recommended per session in the specific device used for the tests, it is apparent that the maximum number of clients per session should be 15, since that is still when the transport is delivering full-sized messages and there is no performance degrading that relies on the optimization algorithm. In a real environment, there is the possibility of packet-drop, duplication and even tampering, so by having a maximum of 15 users, the data usage is not only more predictable and linear but still takes advantage of the KCP optimization algorithm if needed. It should be taken into consideration that the performance of the KCP transport depends on several factors, like the hardware resources, network conditions and implementation efficiency, and these tests are dependent on the device used for the research.

### 5.1.4 Latency and FPS Tests

On the desktop, the application's FPS was almost always the maximum 60 FPS, however, there were small stutters, especially when changing scenes. On the other hand, when deploying the application on the HoloLens 2, the average FPS was very low, ranging from 8 to 12 FPS on normal activities inside the session. Even though this could not be entirely fixed, since the origin of the issue is not completely known, after changing the graphics resolution from High to Low, the average FPS was now ranging from 22 to 30 FPS. It is suspected that the problem derives from the high-complex model of the robot and not any kind of networking issue, since an application without the 3D model loaded yet would reach the expected minimum of 30 FPS. These values were obtained using Unity Profiler and the in-game MRTK Visual Profiler Interface, which displays the FPS to the user in real-time.

Finally, it was also tested the latency and response-delay depending on the number of users joining a specific session, as seen in Fig. 5.13. As expected, while there is a small number of users, the average RTT is a low value - around 30ms -, but as the number of clients expands the user ping also grows. It is of note that the trend line traced does not equate to a clear-cut value because the relationship between the number of users and RTT is not strictly linear or exponential. It is only dependent on the network infrastructure, the efficiency of the server-client communication and its respective quality. On the graph, it is noticeable that the RTT is quite variable, which is the result of the load optimization algorithm of the KCP transport in action. But at a higher number of users, the network can not withstand so many users, even while optimizing the packets trades, this is because of the congestion created and the hardware used for the network infrastructure. In a dedicated server, this could be mitigated by distributing the application server infrastructure across multiple smaller edge servers that still synchronize to a main server, however, since the number of expected users is not higher than 15, the RTT value is reasonably acceptable - lower than 100ms in average.

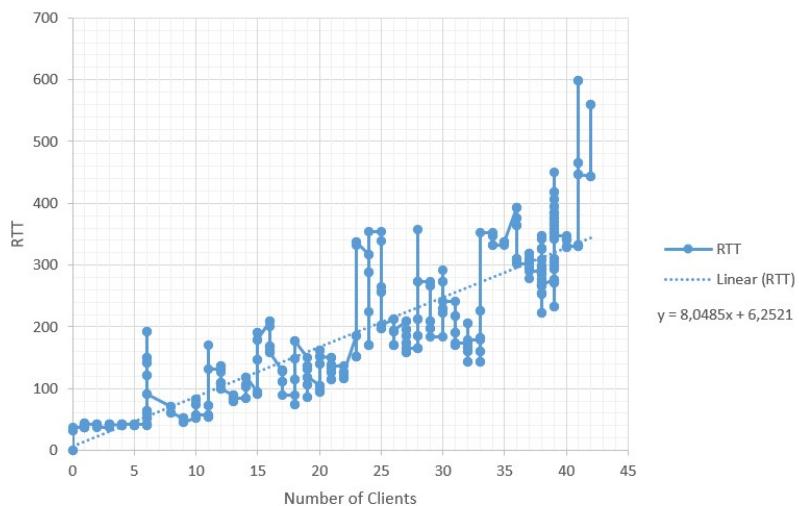


Figure 5.13: Client RTT per Client Instance in a Session

## 5.2 Open Survey on UI/UX Demo

After developing a working demo, there was a need to know if the UI would get in the way of the collaborative process between users or if their personal experience was as maximized as possible. Since the gathering of user feedback plays a crucial role in the evaluation and future refinement of the application, and the success of the application greatly depends on user satisfaction and engagement, the demo was made public online with an open form for users to give their opinion. Developing the single-player demo meant reworking some scripts to simulate the database interaction and deactivating every network component, for instance, if the user created a session, it should still appear on the session list; however, the user would not have access to a database, so plenty of back-end logic had to be restructured. The demo was developed for the WebGL platform and was hosted on Unity Play, [simmer.io](https://simmer.io/)<sup>4</sup> and [gamepipe.io](https://gamepipe.io/)<sup>5</sup>. The demo and the form were published on online forums mainly related to 3D Unity, Augmented Reality, Robotics, Software Development, and Engineering. Since this was a completely anonymous online form where the only method of user authentication was through the obligatory use of the user's personal e-mail, the survey answers should not be taken as absolute truth and instead, be used as guiding points for possible problems. It is also relevant to assess that the application was developed for AR and not web, and these results are to be used as comparative and correlative conclusions, especially when considering that the navigation and interaction system on the web-version is significantly less intuitive than on the HoloLens2.

In total, there are 42 valid answers, although the demo had a total of 93 online plays. According to Janet Six and Ritch Macefield [74], the minimum recommended number of participants per iteration is 5, but there was a huge expected level of problem discovery when interviewing 10 participants. They noticed an effect where the additional impact that the number of participants had on the valuable feedback decreased exponentially with the growing number of users. Considering this, having over 40 valid participants meant that the feedback would be more than enough to evaluate the UI/UX. The form and the answers can be found in the Appendix of the document. The survey was divided into six main sections:

- Section 1 - User Profiling - The first part was used to briefly explain the test process and context and how long it should take to answer all questions. It was also utilized to obtain basic demographic information about the user, like gender, age and occupation. Plus, a pledge was made, where the user commits to giving honest answers for the sake of the study.
- Section 2 - User Authentication - After giving some information on how they should interact on the demo, since the inputs are different from the usual 3D game, the minimum information was given to see if the user could pass the test intuitively. A simple login and password were given and at the end, on the form, the user would evaluate how intuitive the login page was from 1 to 10 and, if needed, give extra feedback on a written response.

---

<sup>4</sup><https://simmer.io/>

<sup>5</sup><https://gamepipe.io/>

- Section 3 - Session Creation - Even though the user is not only completely free to explore the demo, it is also encouraged to do so. After logging in, he would reach one of the two UI menus, the session creation or the session list which was empty, so the user would naturally lurk to the one with actual interactable content. Once again, not much information on the situation is given, just enough for the user to know that creating a server is one of the tasks to be evaluated. They have the ability to tinker with the inputs and then give their feedback again.
- Section 4 - Joining Session - After creating the session, the most natural response should be to check if the session they created was there, which should be. They are prompt to join the session and then give their opinion on the UI and how smooth the flow between menus was, as in, if it was intuitive to create or join sessions.
- Section 5 - Robot Manipulation and Session Menu Page - It is given some information on how to interact with the fake, manipulable robot via grabbing the small sphere - the *Target*. Once again, there is reassurance for them to play with the demo and see how it reacts, especially by utilizing the session menu UI. Then, their feedback on how intuitive it is to use the robot and the session UI is taken once again.
- Session 6 - Final Evaluation - The final section asks for their overall opinion on the UI/UX, if they could find the intended features with ease, a short text response for them to fill if they fill like adding feedback on something, and finally they are asked to describe the UI using some predetermined terms. These terms are meaningful for the UI design iteration, and what the next iteration should be focused on.

For the user profiling, the objective was to know if the user that took the test was experienced with technology and what kind of occupation they had. Even though it was an anonymous answer, it is possible to differentiate the answers through the age, gender and the kind of job they have. This could be useful for multiple reasons, for instance, a professional engineer could take a look into the application and instantly evaluate possible problems with the implementation, while on the other hand, a person that has lower-to-none experience would rate a more honest value on the UI since their experience would be the closest to a beginner and the application was supposed to work for anybody even if they had never picked it up before. The age ranges were also planned consciously: people with less than 18 years are prone to have lower experience in this environment, people between 18 and 25 are most likely students or newly employed workers, while the folks between 26 and 35 already had plenty of work experience by the time they took the test. It is possible to check the participants' age, gender, and career distribution in Fig. 5.14, Fig. 5.15, and Table 5.1.

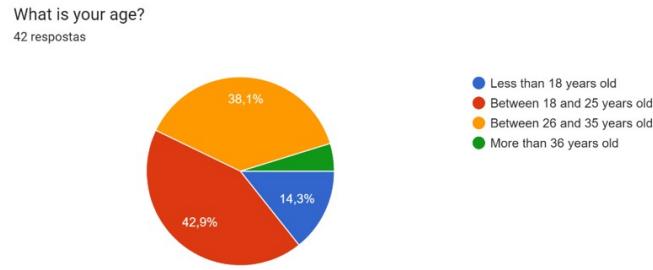


Figure 5.14: Form Testers Age

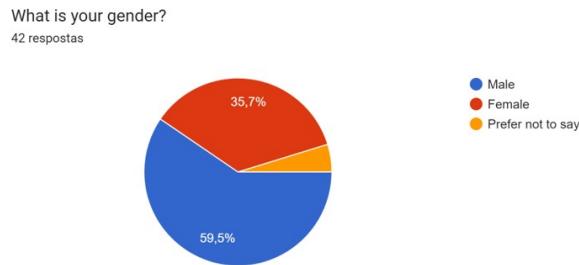


Figure 5.15: Form Testers Gender

Table 5.1: Form Testers Occupation

Occupation	Percentage
Academic	30,9%
Engineering	19%
Programming	16,7%
Other	7,14%
No Answer	26,2%

For Section 2, the overall majority considered the login page remarkably intuitive, as it is possible to see in Fig. 5.16. Some users left some feedback related to it: most of the opinions were on how the keyboard input works oddly, but some people said that the UI lacked some charm or adding a microphone icon could be added to let people know that they are being recorded.

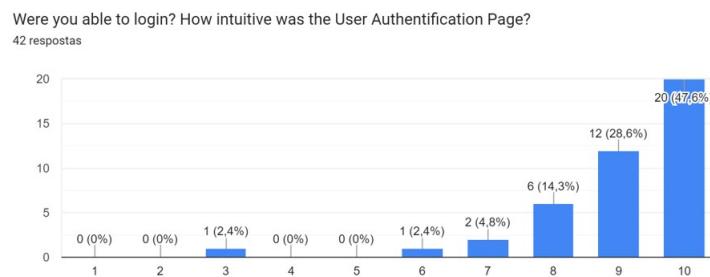


Figure 5.16: Form Testers Opinion on Login Menu UI

In Section 3 it is possible to see a decrease in user satisfaction, but overall it is still mainly positive (Fig. 5.17). The complaints related to the session creation menu are mainly targeted at the need for multiple inputs and displaying too much information on a single screen. Even though this menu went through severe changes through multiple iterations, it still is the menu that raises some user experience issues. The attempted solution to the issues brought up by the user's opinions was to make the robot IP address and port invisible until you toggle the *real robot* button.

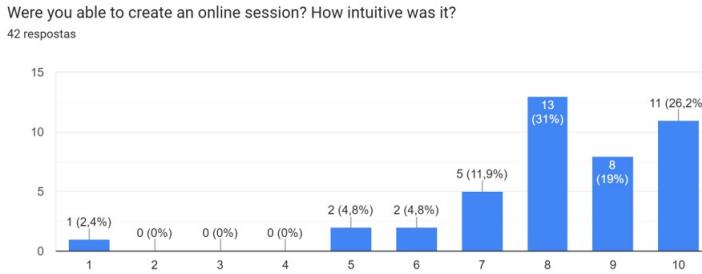


Figure 5.17: Form Testers Opinion on Session Creation Menu UI

Section 4's feedback was overwhelmingly positive, which is a result of having as few visual components on the screen at the same time, which results in a clean-looking UI and simple to navigate (Fig. 5.18 and Fig. 5.19). There were no user comments on this menu. The side menu used for quickly switching back and forth between session creation and session joining seems to also be effective since the responses to it are also rather favorable.

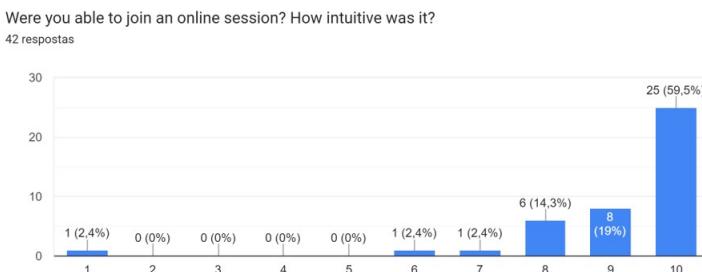


Figure 5.18: Form Testers Opinion on Joining Session Menu UI

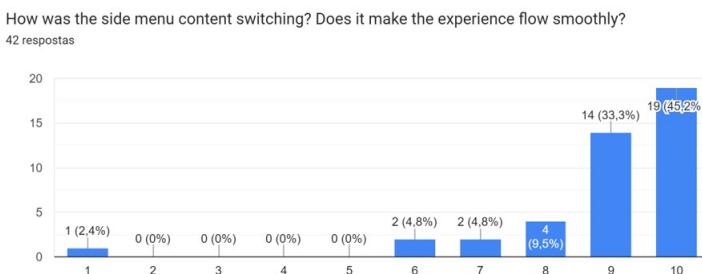


Figure 5.19: Form Testers Opinion on Side Menu UI Switching

For Section 5 it is possible to see that robot handling was not something native or naturally intuitive for them, but in the end, it still unequivocally served its job and people understood that (Fig. 5.20). On the other hand, the Online Session UI seemed to be more controversial, where most people gave more diverse feedback (Fig. 5.21). The reason for this may be that the context of the features was not explained in detail, so for some testers, the buttons' purpose could be more difficult to figure out. But overall, the participant's opinion on the UI is still very positive.

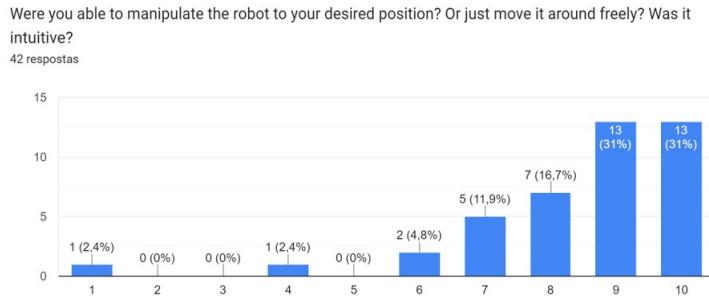


Figure 5.20: Form Testers Opinion on Robot Manipulation

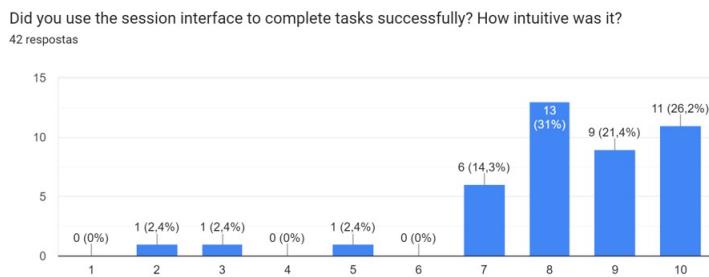


Figure 5.21: Form Testers Opinion on Online Session Menu UI

The final Section 6 asks the testers for their overall input on how intuitive the UI felt and their experience with the application (Fig. 5.22). The predominant opinion is remarkably favorable, even though some users left negative feedback. Most of the users that left their opinion asked to change specific icons on the UI that were confusing or unappealing, for instance. The UI's difficulty seems to also be low, according to the user's grading, which means that the interface should be optimal to complete beginners (Fig. 5.23). It is also possible to evaluate the general opinion on the interface using the table Fig. 5.2 as reference. The term most chosen to define the UI was "Easy to Learn" and "Clear" which were the main objectives and design concepts proposed on the UI planning process in Section 4.2. It is possible to conclude through the form's responses that the application had a welcoming design, with a professional outlook, and was precisely suited for industry workers, especially since applying this web-based application into an AR system would significantly upgrade the UI interaction and navigation.

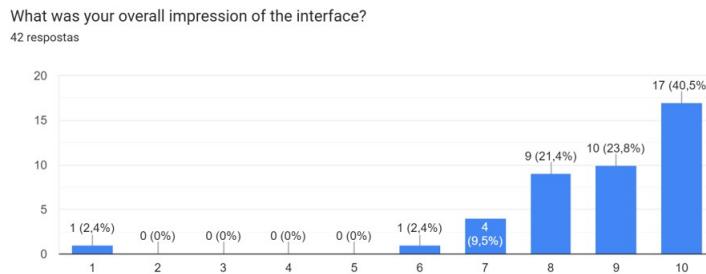


Figure 5.22: Form Testers Final Opinion on the UI

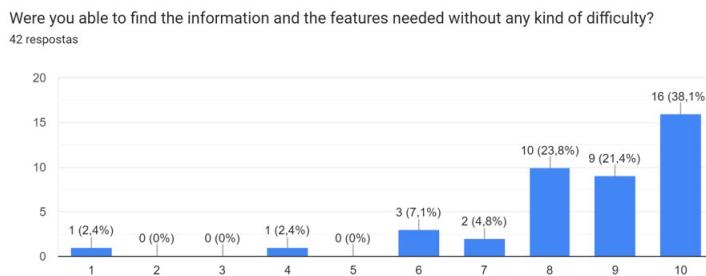


Figure 5.23: Form Testers Opinion on the Difficulty of the UI

Table 5.2: Form Testers Opinion on UI

Term	Percentage
Easy to Learn	81%
Clear	78,6%
Simple to Grasp	69%
User-Friendly	57,1%
Efficient	54,8%
Well Organized	45,2%
Interactive	35,7%
Intuitive	28,6%
Cohesive	28,6%
Consistent	28,6%
Engaging	21,4%
Responsive	14,3%

# **Chapter 6**

## **Conclusions and Future Work**

### **6.1 Conclusions**

The main objective of this dissertation was to expand on the HMI paradigm by introducing a multiuser system through the use of AR. By promoting an enhanced interaction between multiple humans and machinery in an online session it would be possible to solve some of the issues that Industry 4.0 arises, like the necessity of swift and cheaper training and the ability to solve problems faster via collaborative participation.

As such, several topics and technologies related to the dissertation's theme are thoroughly investigated and scrutinized in order to understand the requirements of a possible solution and how the respective implementation shall be done. Firstly, it was analyzed how the AR technology functions and its strive for modern-day applications; then the interaction between a human and a machine is closely surveyed in order to achieve a high-level collaborative experience. It is examined the employment and design of the interface acting as the intermediate between the machine and the operator, and the distinct variations pertaining to it. Then, multiuser system solutions are researched and how they could significantly improve the collaborative process, while also analyzing specific employments of multiuser implementations in an AR environment.

After analyzing and reviewing the literature and giving some background on the concepts related to the project, the theoretical design of the system is made by analyzing some of the challenges that the technologies impose, and also the requirements for the system planning are examined taking into consideration the research made. Each technological component is thoroughly analyzed in order to not only understand and conceptualize how these frameworks could be consolidated into the final system but what kind of prerequisites the architecture would need. After outlining some compelling features that the system could incorporate and how those components would interact between themselves by outlining some UML diagrams and use-cases, a final architecture is proposed. Given the technologically visionary nature of the project, it is crucial to prioritize a scalable and flexible architecture. By anticipating future growth and ensuring the system can accommodate those demands, the system should be able to remain adaptable enough to support future advancements.

The planned system should present the possibility and characteristics of an Augmented Reality interface that permits the user to see the real world with virtual information overlayed. By integrating AR with an HMI system, it should enable some noteworthy interaction between users and machines, taking intuitive and natural cooperation to a whole new level by exploiting the reality that the user is already familiar with. By gripping and holding a virtual model of the robot, the user should be able to move and orient the machine due to the consolidation between the robot configuration and its digital twin. This interaction should feel intuitive and easy to perceive for the operator due to the human-based nature of gesture control. The robot configuration is performed through the use of the framework ROS. Furthermore, the possibility of multiple users controlling the machine simultaneously, with the implementation of specific multiuser protocols and a reliable network protocol, could not only enhance the sense of collaboration between teams but also boost the overall efficiency of manufacturing by enabling the concurrent execution of tasks.

After actually implementing and deploying the system, multiple tests were made in order to validate the project's capabilities. The experiments were divided into two main sections: the first was stress-testing the network to its maximum potential and the other was evaluating the application's user experience. Stress-testing a multiuser application is essential to ensure its optimal performance and reliability in real-world scenarios. By subjecting the network to simulated high user loads and intense usage events, these tests help identify potential bottlenecks and find areas that are in need of fine-tuning. Additionally, it is possible to estimate the optimal environment in which the system could be thriving. The second test relied on receiving feedback from users that experimented with the application. By actively seeking and gathering user feedback, it is possible to gain outside insight into the application's usability, functionality, and overall user experience. It also helps to identify features that are lacking and in need of improvement, uncover issues, and also understand user needs and preferences. Both tests were noticeably positive and the project can be considered complete.

Although, it is worth bearing in mind that the actual integration of the project into the industry is not absolutely direct: there is still the need for tweaks and possible upgrades to the implementations. There are hard-coded elements that would need to be manually modified, the master server - even though it serves its purpose for now - in the future, it would need to be customized to the specific industry. Overall, the project would need to be tailored to fit in with the specific manufacturing industry.

## **6.2 Future Work**

As previously said, even though the mainframe of the system is complete and works as expected, first some of the features should be ironed out, since there is still some known software issues. The system was only implemented for the UR10 robot, and there are plenty of other collaborative robots that could be incorporated into the project, but this is not a big issue: by copying the structure of the programming infrastructure but uploading a different robot model via URDF should work as expected.

Then there is the real robot interaction, which was not the main focus of the project. Even though the connection to the real robot was established and a ROS Publish function was tested and validated, while working on the project, the iiLab team further developed the *Programming* functionality which now has a whole new UI with plenty of features. The programming function implemented should be used as a template tool to implement the rest of the functionalities, but overall, this is rather feasible.

Also, there is the master server that needs to be customized to suit the ever-growing number of *Online Scenes* with different robot models which can be fake or real. Additionally, in order to debug a conflict, the input argument passed for the server launch that had the robot IP address appended had to be removed - although with more time this could be actually fixed with no need for a workaround.

In the future, the website can be further developed in order to accommodate more features, such as allowing users to create or list all online sessions through the web, display a history of their online work, or add the possibility to check the machinery or manufacturing processes' status. This could also be helpful for remote monitoring of the robots and providing real-time assistance to workers.

Moreover, there is still the marker-based deprecated feature that should eventually be re-introduced. It does not have to be necessarily a marker-based program, since markerless algorithms are now being further developed and promoted. Also, with the final implementation of the system, it is possible to join sessions remotely, from any other location than the local area network (LAN), through the use of a VPN to connect to the LAN. But new ways of interacting between users need to be implemented, like voice calls.

With the aim of researching multiuser collaboration in HMI for Industry 4.0, more social user tests should be made. This way it is possible to take in the real-experience feedback and focus on features that seem to matter the most to improve the application. In this dissertation, only the UI and UX is evaluated, not the multiuser general experience.

Finally, as the project integrates ambitious technologies, there could be some more future compelling implementations capable of enhancing and expanding the capabilities of the system, such as cloud computing to store and process large amounts of data, edge computing, for a seamless remote connection from anywhere around the globe, Internet of Things (IoT) to connect the robots to the IoT ecosystem, allowing for more automation and better coordination between systems. To leverage a high-speed connection and take advantage of the almost real-time, ultra-low latency communication, the integration of 5G would significantly upgrade the network.



# Appendix A

## Public UI Demo Test

### A.1 Public UI Demo Test Form

#### Thesis - Collaborative Human Machine Interface through Augmented Reality

Hey! I'm reaching out to you and this community with a survey to study the efficiency of the User Interface created for my Master's in Electrical and Computer Engineering on the University of Porto. I'm asking for help on this online community in specific because it either has either a vast experience on high-end technologies or a very intrinsic interest on matters like these.

A little background on the aim of the study: This interface is going to be used in an industrial and collaborative background in order to control virtually, through Augmented Reality, the movement of Collaborative Robots (Cobots) that help you with your tasks, making them faster and easier to perform. The big emphasis on this application is the possibility to work simultaneously with other people on the same robot, so they can function together towards the same objective, making it a process of communication, learning and adapting. However, since this is just a UI user test, the collaborative features are not implemented on purpose. We just want to see how intuitive and simple the interface is/can become.

Thank you for participating!! You should launch the demo application to mess around with it. The test and the application will be available for a short period of time. Also, every data you send is obviously confidential and the privacy of it is assured, so we ask for you to be as unbiased as possible while taking the test. It should take less than 10 minutes to complete the survey! And there are no right or wrong answers!

Sincerely,  
João Peixoto

\* Indica uma pergunta obrigatória

1. Email \*

2. What is your age? \*

Marcar apenas uma oval.

- Less than 18 years old
- Between 18 and 25 years old
- Between 26 and 35 years old
- More than 36 years old

3. What is your gender? \*

Marcar apenas uma oval.

- Male
- Female
- Prefer not to say

4. What is your occupation?

5. Do commit to give fully honest answers for the sake of the integrity of the study? \*

Marcar apenas uma oval.

- Yes
- No

## Task 1 - User Authentication

**Just a small tutorial on how to interact with the application.** Since this is developed for AR, you can either: use the small mouse-like button at the middle of the screen, or press the SPACE bar to generate a virtual hand. With the virtual hand, if you click the left mouse button, you can generate a click, or you can either use the mouse scroll and WASD keys to move the hand to your like if you want to tap a button physically instead. Additionally, buttons also have microphone integration: if you focus on a button with the middle icon and say the word that appears under it, it also activates.

Just a simple way of authenticating the user. Go ahead and try to fabricate usernames or passwords. If you really want to login, try using the User ID "Guest1" and password "Guest1". Otherwise you can just press the button "Enter as Guest".

## Authentication Interface



6. Were you able to login? How intuitive was the User Authentication Page? \*

*Marcar apenas uma oval.*

Not intuitive

- 1   
2   
3   
4   
5   
6   
7   
8   
9   
10

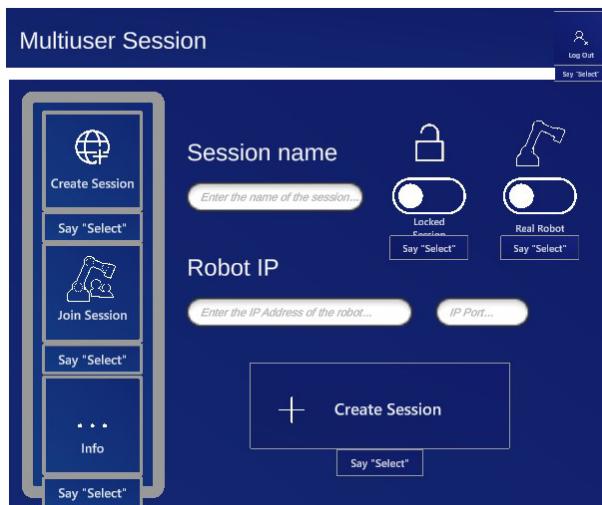
Very intuitive

7. If you have any opinion on the interface, please share your feedback. (i.e. too complex, needs new icons, images, more descriptive text, better design...)
-

## Task 2 - Create a Server

Even though there is an input for the session name, you can leave it as default. There's the option to create a server connected to a real robot by entering its IP and port, or just use a simulated one. You also have the chance of locking the session to outside users, where they must ask for permission to the session owner to connect. After creating the session, in the main application you would be redirected automatically to the session, however, for the sake the test it doesn't have a legit output.

## Create Session Interface



8. Were you able to create an online session? How intuitive was it? \*

*Marcar apenas uma oval.*

Not intuitive



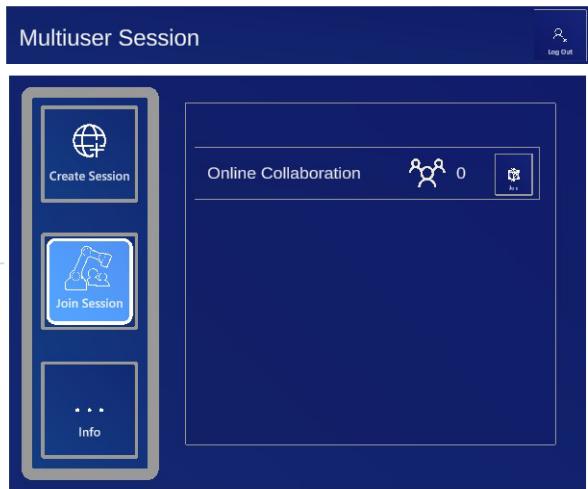
Very intuitive

9. If you have any opinion on the interface, please share your feedback. (i.e. too complex, needs new icons, images, more descriptive text, better design...)
-

**Task 3 - Join a Server**

There should be multiple sessions listed with the number of players visible. These are simulated players just for the sake of the experiment. You can pick any one of the sessions to join it.

Join Session Interface



10. Were you able to join an online session? How intuitive was it? \*

*Marcar apenas uma oval.*

Not intuitive



Very intuitive

11. How was the side menu content switching? Does it make the experience flow \* smoothly?

*Marcar apenas uma oval.*

Bad experience

1  

---

2  

---

3  

---

4  

---

5  

---

6  

---

7  

---

8  

---

9  

---

10  

---

Good experience

12. If you have any opinion on the interface, please share your feedback. (i.e. too complex, needs new icons, images, more descriptive text, better design...)

---

---

---

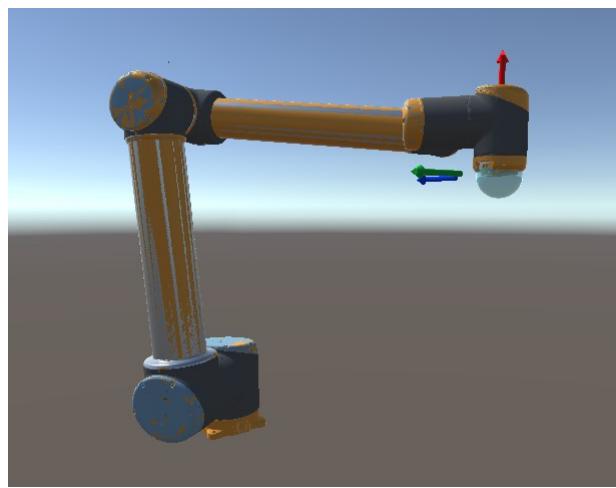
---

#### Task 4 - Robot Manipulation & Session Menu Page

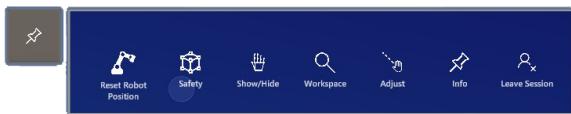
This is the main scene of the session, where you interact with a robot's digital twin (DT) and collaborate with a supervisor/worker to achieve the collaborative task. At the tip of the robot there is a small sphere (Target), you can grab the Target and mess around with the robot's positioning.

Additionally there should be a menu that follows you around: this interface is used to mainly control the session. It has multiple functions: reset the DT's positioning or move it around, create a safety zone or a workspace zone around the robot and, finally, leave the session.

Digital Twin of Collaborative Robot



Session Menu Interface



13. Were you able to manipulate the robot to your desired position? Or just move it around freely? Was it intuitive?

*Marcar apenas uma oval.*

Not intuitive

1

2

3

4

5

6

7

8

9

10

Very intuitive

14. Did you use the session interface to complete tasks successfully? How intuitive was it?

*Marcar apenas uma oval.*

Not intuitive

---

1

2

3

4

5

6

7

8

9

10

---

Very intuitive

---

**Final evaluation**

Just give us your final thoughts on the UI/UX you experienced right now.

15. What was your overall impression of the interface? \*

*Marcar apenas uma oval.*



16. Were you able to find the information and the features needed without any kind \*  
of difficulty?

*Marcar apenas uma oval.*

Hard to use

1 

2 

3 

4 

5 

6 

7 

8 

9 

10 

Easy to use

17. If your answer is below a 5, or if you feel like sharing an opinion, please describe  
your thoughts.

18. Would you use any of this terms to describe the UI? \*

*Marcar tudo o que for aplicável.*

- Easy to learn
- Clear
- Well organized
- Simple to grasp
- Intuitive
- User-Friendly
- Responsive
- Engaging
- Interactive
- Cohesive
- Consistent
- Efficient

Thank you for participating!

Once again, your opinion will be completely anonymous and only used for improvement of this project.

**Thank you!**

João Peixoto

Este conteúdo não foi criado nem aprovado pela Google.

Google Formulários



A.2 Public UI Demo Test Form Responses

# Bibliography

- [1] E. Hofmann and M. Rüscher, “Industry 4.0 and the current status as well as future prospects on logistics,” *Computers in industry*, vol. 89, pp. 23–34, 2017.
- [2] M. Premm and S. Kirn, “A multiagent systems perspective on industry 4.0 supply networks,” in *German Conference on Multiagent System Technologies*, Springer, 2015, pp. 101–118.
- [3] J. Egger and T. Masood, “Augmented reality in support of intelligent manufacturing—a systematic literature review,” *Computers & Industrial Engineering*, vol. 140, p. 106 195, 2020.
- [4] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, “Industry 4.0,” *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.
- [5] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, “Augmented reality: A class of displays on the reality-virtuality continuum,” in *Telemanipulator and telepresence technologies*, International Society for Optics and Photonics, vol. 2351, 1995, pp. 282–292.
- [6] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic, “Augmented reality technologies, systems and applications,” *Multimedia tools and applications*, vol. 51, no. 1, pp. 341–377, 2011.
- [7] D. Van Krevelen and R. Poelman, “A survey of augmented reality technologies, applications and limitations,” *International journal of virtual reality*, vol. 9, no. 2, pp. 1–20, 2010.
- [8] B. Furht, *Handbook of augmented reality*. Springer Science & Business Media, 2011.
- [9] P. Thomas and W. David, “Augmented reality: An application of heads-up display technology to manual manufacturing processes,” in *Hawaii international conference on system sciences*, ACM SIGCHI Bulletin New York, NY, USA, vol. 2, 1992.
- [10] G. Evans, J. Miller, M. I. Pena, A. MacAllister, and E. Winer, “Evaluating the microsoft hololens through an augmented reality assembly application,” in *Degraded environments: sensing, processing, and display 2017*, International Society for Optics and Photonics, vol. 10197, 2017, p. 101970V.
- [11] M. Tezer, E. Yıldız, A. Masalimova, A. Fatkhutdinova, M. Zheltukhina, and E. Khairullina, “Trends of augmented reality applications and research throughout the world: Meta-analysis of theses, articles and papers between 2001-2019 years,” *International Journal of Emerging Technologies in Learning (iJET)*, vol. 14, no. 22, pp. 154–174, 2019.

- [12] T. Starner, S. Mann, B. Rhodes, J. Healey, K. B. Russell, and A. Pentland, “Wearable computing and augmented reality,”
- [13] C. Bichlmeier, F. Wimmer, S. M. Heinig, and N. Navab, “Contextual anatomic mimesis hybrid in-situ visualization method for improving multi-sensory depth perception in medical augmented reality,” in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, IEEE, 2007, pp. 129–138.
- [14] M. Morozov, *Augmented reality in military: Ar can enhance warfare and training*. [Online]. Available: <https://jasoren.com/augmented-reality-military/>, Accessed on Feb. 21, 2022.
- [15] *Augmented reality in healthcare*. [Online]. Available: <https://southgatemedical.com.au/augmented-reality-in-healthcare/>, Accessed on Feb. 21, 2022.
- [16] M. K. Bekele, R. Pierdicca, E. Frontoni, E. S. Malinverni, and J. Gain, “A survey of augmented, virtual, and mixed reality for cultural heritage,” *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 11, no. 2, pp. 1–36, 2018.
- [17] I. P. Tussyadiah, T. H. Jung, and M. C. tom Dieck, “Embodiment of wearable augmented reality technology in tourism experiences,” *Journal of Travel research*, vol. 57, no. 5, pp. 597–611, 2018.
- [18] D. Yu, J. S. Jin, S. Luo, W. Lai, and Q. Huang, “A useful visualization technique: A literature review for augmented reality and its application, limitation & future direction,” *Visual information communication*, pp. 311–337, 2009.
- [19] C. Perey, “Augmented reality for basel,” 2011. [Online]. Available: <https://www.perey.com/AugmentedRealityForBasel/>.
- [20] J. H. Shuhaimer, “Augmented reality in surgery,” *Archives of surgery*, vol. 139, no. 2, pp. 170–174, 2004.
- [21] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, “Recent advances in augmented reality,” *IEEE computer graphics and applications*, vol. 21, no. 6, pp. 34–47, 2001.
- [22] Q. Ke, J. Liu, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, “Computer vision for human–machine interaction,” in *Computer Vision for Assistive Healthcare*, Elsevier, 2018, ch. 5, pp. 127–145.
- [23] E. Bottani and G. Vignali, “Augmented reality technology in the manufacturing industry: A review of the last decade,” *IIEE Transactions*, vol. 51, no. 3, pp. 284–310, 2019.
- [24] A. Y. Nee and S.-K. Ong, “Virtual and augmented reality applications in manufacturing,” *IFAC proceedings volumes*, vol. 46, no. 9, pp. 15–26, 2013.
- [25] F. Doil, W. Schreiber, T. Alt, and C. Patron, “Augmented reality for manufacturing planning,” in *Proceedings of the workshop on Virtual environments 2003*, 2003, pp. 71–76.

- [26] Y. Shen, S. K. Ong, and A. Y. Nee, “Augmented reality for collaborative product design and development,” *Design studies*, vol. 31, no. 2, pp. 118–145, 2010.
- [27] W. Fang and Z. An, “A scalable wearable ar system for manual order picking based on warehouse floor-related navigation,” *The International Journal of Advanced Manufacturing Technology*, vol. 109, no. 7, pp. 2023–2037, 2020.
- [28] M. Leonard, “Pick speed improves when augmented reality replaces paper: Study,” Jan. 2021. [Online]. Available: <https://www.supplychaindive.com/news/picker-augmented-reality-speed-warehouse-ar-china/593989/>.
- [29] A. Y. Nee, S. Ong, G. Chryssolouris, and D. Mourtzis, “Augmented reality applications in design and manufacturing,” *CIRP annals*, vol. 61, no. 2, pp. 657–679, 2012.
- [30] S. K. Ong and A. Y. C. Nee, *Virtual and augmented reality applications in manufacturing*. Springer Science & Business Media, 2013.
- [31] O. Bimber and R. Raskar, *Spatial augmented reality: merging real and virtual worlds*. CRC press, 2005.
- [32] A. Doshi, S. Y. Cheng, and M. M. Trivedi, “A novel active heads-up display for driver assistance,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 85–93, 2008.
- [33] J. R. Vallino, *Interactive augmented reality*. University of Rochester, 1998.
- [34] R. Raskar, G. Welch, and H. Fuchs, “Spatially augmented reality,” *Augmented Reality: Placing Artificial Objects in Real Scenes*, pp. 64–71, 1999.
- [35] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International conference on computer vision*, Ieee, 2011, pp. 2564–2571.
- [36] H. I. Cakir, B. Benligiray, and C. Topal, “Combining feature-based and model-based approaches for robust ellipse detection,” in *2016 24th European signal processing conference (EUSIPCO)*, IEEE, 2016, pp. 2430–2434.
- [37] R. T. Azuma, “A survey of augmented reality,” *Presence: teleoperators & virtual environments*, vol. 6, no. 4, pp. 355–385, 1997.
- [38] T. Luhmann, “Precision potential of photogrammetric 6dof pose estimation with a single camera,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 3, pp. 275–284, 2009.
- [39] C. Nass, J. Steuer, and E. R. Tauber, “Computers are social actors,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1994, pp. 72–78.
- [40] H. D. Unbehauen, *CONTROL SYSTEMS, ROBOTICS AND AUTOMATION–Volume XXI: Elements of Automation*. EOLSS Publications, 2009.
- [41] T. M. Newcomb, R. H. Turner, and P. E. Converse, *Social psychology: The study of human interaction*. Psychology Press, 2015.

- [42] J. N. Bailenson and N. Yee, "Digital chameleons: Automatic assimilation of nonverbal gestures in immersive virtual environments," *Psychological science*, vol. 16, no. 10, pp. 814–819, 2005.
- [43] F. Ansari, S. Erol, and W. Sihn, "Rethinking human-machine learning in industry 4.0: How does the paradigm shift treat the role of human learning?" *Procedia manufacturing*, vol. 23, pp. 117–122, 2018.
- [44] Y. Inagaki, H. Sugie, H. Aisu, S. Ono, and T. Unemi, "Behavior-based intention inference for intelligent robots cooperating with human," in *Proceedings of 1995 IEEE International Conference on Fuzzy Systems.*, IEEE, vol. 3, 1995, pp. 1695–1700.
- [45] C. Bolton, V. Machová, M. Kovacova, and K. Valaskova, "The power of human–machine collaboration: Artificial intelligence, business automation, and the smart economy," *Economics, Management, and Financial Markets*, vol. 13, no. 4, pp. 51–56, 2018.
- [46] A. Dattalo, "Iso/TS 15066:2016, Robots and robotic devices - collaborative robots," Feb. 2016.
- [47] A. Vysocky and P. Novak, "Human-robot collaboration in industry," *MM Science Journal*, vol. 9, no. 2, pp. 903–906, 2016.
- [48] C. J. Conti, A. S. Varde, and W. Wang, "Robot action planning by commonsense knowledge in human-robot collaborative tasks," in *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, IEEE, 2020, pp. 1–7.
- [49] A. Kolbeinsson, E. Lagerstedt, and J. Lindblom, "Classification of collaboration levels for human-robot cooperation in manufacturing," in *Advances in Manufacturing Technology XXXII*, IOS Press, 2018, pp. 151–156.
- [50] D. Vernon, *Artificial cognitive systems: A primer*. MIT Press, 2014.
- [51] L. Gualtieri, E. Rauch, and R. Vidoni, "Emerging research fields in safety and ergonomics in industrial collaborative robotics: A systematic literature review," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 101998, 2021.
- [52] P. Dillenbourg and D. Schneider, "Collaborative learning and the internet," 1995.
- [53] E. Peters, B. Heijligers, J. de Kievith, *et al.*, "Design for collaboration in mixed reality: Technical challenges and solutions," in *2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*, IEEE, 2016, pp. 1–7.
- [54] S. C. Kendall, J. Waldo, A. Wollrath, and G. Wyant, *A note on distributed computing*, 1994.
- [55] W. A. Hatem, A. Kwan, and J. Miles, "Comparing the effectiveness of face to face and computer mediated collaboration," *Advanced Engineering Informatics*, vol. 26, no. 2, pp. 383–395, 2012.
- [56] A. Henrysson, M. Billinghamst, and M. Ollila, "Face to face collaborative ar on mobile phones," in *Fourth ieee and acm international symposium on mixed and augmented reality (ismar'05)*, IEEE, 2005, pp. 80–89.

- [57] S. Gauglitz, B. Nuernberger, M. Turk, and T. Höllerer, “World-stabilized annotations and virtual scene navigation for remote collaboration,” in *Proceedings of the 27th annual ACM symposium on User interface software and technology*, 2014, pp. 449–459.
- [58] M. Karrer, P. Schmuck, and M. Chli, “Cvi-slam—collaborative visual-inertial slam,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2762–2769, 2018.
- [59] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli, “Covins: Visual-inertial slam for centralized collaboration,” in *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, IEEE, 2021, pp. 171–176.
- [60] D. Schmalstieg, A. Fuhrmann, and G. Hesina, “Bridging multiple user interface dimensions with augmented reality,” in *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, IEEE, 2000, pp. 20–29.
- [61] W. Zhang, B. Han, and P. Hui, “Sear: Scaling experiences in multi-user augmented reality,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 5, pp. 1982–1992, 2022.
- [62] M. Ostanin, S. Mikhel, A. Evlampiev, V. Skvortsova, and A. Klimchik, “Human-robot interaction for robotic manipulator programming in mixed reality,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 2805–2811.
- [63] I. Soares, M. Petry, and A. P. Moreira, “Programming robots by demonstration using augmented reality,” *Sensors*, vol. 21, no. 17, p. 5976, 2021.
- [64] T. Inamura and Y. Mizuchi, “Sigverse: A cloud-based vr platform for research on social and embodied human-robot interaction,” *arXiv preprint arXiv:2005.00825*, 2020.
- [65] Y. Mizuchi and T. Inamura, “Cloud-based multimodal human-robot interaction simulator utilizing ros and unity frameworks,” in *2017 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2017, pp. 948–955.
- [66] K. Yu, U. Eck, F. Pankratz, M. Lazarovici, D. Wilhelm, and N. Navab, “Duplicated reality for co-located augmented reality collaboration,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 5, pp. 2190–2200, 2022.
- [67] G. Gardiner, *Future of design with microsoft hololens and fusion 360*, Nov. 2015. [Online]. Available: <https://www.autodesk.com/products/fusion-360/blog/future-of-design-with-microsoft-hololens-and-fusion-360/>, Accessed on Jan. 5, 2022.
- [68] A. Coppens, “Merging real and virtual worlds: An analysis of the state of the art and practical evaluation of microsoft hololens,” *arXiv preprint arXiv:1706.08096*, 2017.
- [69] M. Quigley, K. Conley, B. Gerkey, *et al.*, “Ros: An open-source robot operating system,” in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

- [70] M. Bischoff, “Ros # on github.com/siemens/ros-sharp,” Dec. 2017. [Online]. Available: <https://discourse.ros.org/t/ros-on-github-com-siemens-ros-sharp/3405>.
- [71] K. Semple, Q. Wen, and K. Sharkey, “What is the mixed reality toolkit,” Nov. 2021. [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05>, Acessed on Feb. 11, 2022.
- [72] B. House, “How to choose the right netcode for your game,” Sep. 2020. [Online]. Available: <https://blog.unity.com/games/how-to-choose-the-right-netcode-for-your-game>.
- [73] C. Heer, “Robot density nearly doubled globally,” *International Federation of Robotics reports*, Dec. 2021. [Online]. Available: <https://ifr.org/ifr-press-releases/news/robot-density-nearly-doubled-globally>.
- [74] J. M. Six and R. Macefield, “How to determine the right number of participants for usability studies,” *San Francisco (CA): UXmatters*, 2016.